Project Acronym: Open-DAI

Grant Agreement number: 297362

Project Title: Opening Data Architectures and Infrastructures of European Public Administrations

Work Package: Pilot development

Deliverable Number: D5.2

Revision History

Revision Date	Author	Organisation	Description
1 Jul 2013	Marc Planagumà	BDigital	Draft ver. 0.1
9 Oct 2013	Marc Planagumà	BDigital	Final ver. 1.0
17 Mar 2014	Marc Planagumà	BDigital	Revision ver. 2.0
6 Jun 2014	Marc Planagumà	BDigital	Revision ver. 2.1
29 Jun 2014	Marc Planagumà	BDigital	Revision ver. 2.2
11 Jul 2014	Marc Planagumà	BDigital	Final Reviewed ver 2.3

Legal Disclaimer

Copyright 2013 by CSI-Piemonte, BDIGITAL, SAMPAS, NetPort, Regione Piemonte, Karlshamn Municipality, Ordu Municipality, Barcelona Municipality, Lleida Municipality, Politecnico di Torino, DIGITPA.

The information in this document is proprietary to the following Open-DAI Consortium members: CSI- Piemonte, BDIGITAL, SAMPAS, NetPort, Regione Piemonte, Karlshamn Kommun, Ordu Municipality, Barcelona Municipality, Lleida Municipality, Politecnico di Torino, DIGITPA.

This document contains preliminary information and it is available under the term of the following license:

The Open-DAI Data Assessment and Specification Report by Open-DAI Project is licensed under a Creative Commons Attribution 3.0 Unported License.

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Table of Contents

1 Pilots implementation report 7
1.1 Piedmont Region Pilots implementation 7
1.1.1 TWIST accident publishing 7
1.1.1.1 Pilot development 8
1.1.1.1.1 Followed development plan 8
1.1.1.1.2 Development results 9
1.1.1.2 Pilot documentation: 9
1.1.1.2.1 Pilot code repository 9
1.1.1.2.2 How-to install and deploy 10
1.1.1.2.3 User Manual 10
1.1.2 TWIST real time accident collection/SaveUS 11
1.1.2.1 Pilot development 11
1.1.2.1.1 Followed development plan 12
1.1.2.1.2 Development results 12
1.1.2.2 Pilot documentation: 13
1.1.2.2.1 Pilot code repository 13
1.1.2.2.2 Pilot code guide 13
1.1.2.2.3 How-to install and deploy 14
1.1.2.2.4 User Manual 15
1.1.3 Air Quality 19
1.1.3.1 Pilot development 19
1.1.3.1.1 Followed development plan 20
1.1.3.1.2 Development results 20
1.1.3.2 Pilot documentation: 20
1.1.3.2.1 Pilot code repository 20
1.1.3.2.2 Pilot code guide 20
1.1.3.2.3 How-to install and deploy 21
1.1.3.2.4 User Manual 21
1.1.4 Transport API 23
1.1.4.1 Pilot development 23
1.1.4.1.1 Followed development plan 23
1.1.4.1.2 Development results 23
1.1.4.2 Pilot documentation: 24
1.1.4.2.1 Pilot code repository 24
1.1.4.2.2 How-to install and deploy 24
1.1.4.2.3 User Manual 24
1.2 Barcelona municipality Pilots 26
1.2.1 Pollution monitor for outdoor activities Pilot (Mobile App) 26
1.2.1.1 Pilot development 26
1.2.1.1.1 Followed development plan 28
1.2.1.1.2 Development results 28
1.2.1.2 Pilot documentation: 29
1.2.1.2.1 Pilot code repository 29
1.2.1.2.2 How-to install and deploy 30

1.2.1.2.3 User Manual 31
1.2.2 Correlation between environment and mobility (Web App) 32
1.2.2.1 Pilot development 32
1.2.2.1.1 Followed development plan 34
1.2.2.1.2 Development results 34
1.2.2.2 Pilot documentation: 35
1.2.2.2.1 Pilot code repository 35
1.2.2.2.2 How-to install and deploy 35
1.2.2.2.3 User Manual 36
1.3 Lleida municipality Pilots 38
1.3.1 Road Incidents and Accessibility (Mobile App) 38
1.3.1.1 Pilot development 38
1.3.1.1.1 Followed development plan 40
1.3.1.1.2 Development results 40
1.3.1.2 Pilot documentation: 41
1.3.1.2.1 Pilot code repository 41
1.3.1.2.2 How-to install and deploy 41
1.3.1.2.3 User Manual 42
1.3.2 Data publication (Web App) 44
1.3.2.1 Pilot development 44
1.3.2.1.1 Followed development plan 46
1.3.2.1.2 Development results 46
1.3.2.2 Pilot documentation: 47
1.3.2.2.1 Pilot code repository 47
1.3.2.2.2 How-to install and deploy 47
1.3.2.2.3 User Manual 48
1.4 Ordu municipality Pilots 50
1.4.1 City Dynamics (Mobile App) 50
1.4.1.1 Pilot development 50
1.4.1.1.1 Followed development plan 59
1.4.1.1.2 Development results 60
1.4.1.2 Pilot documentation: 61
1.4.1.2.1 Pilot code repository 61
1.4.1.2.2 How-to install and deploy 61
1.4.1.2.3 User Manual 62
1.4.2 Poi`s (Mobile App) 66
1.4.2.1 Pilot development 66
1.4.2.1.1 Followed development plan 73
1.4.2.1.2 Development results 74
1.4.2.2 Pilot documentation: 75
1.4.2.2.1 Pilot code repository 75
1.4.2.2.2 How-to install and deploy 75
1.4.2.2.3 User Manual 76
1.5 Karlshamn municipality pilot 79
1.5.1 Reports 79
1.5.1.1 Pilot development 80
1.5.1.1.1 Followed development plan 81

1.5.1.1.2 Development results....81
1.5.1.2 Pilot documentation:....82
1.5.1.2.1 Pilot code repository....82
1.5.1.2.2 How-to install and deploy....82
1.5.1.2.3 User Manual....83

List of Figures

Figure 1: Block diagram TWIST accident publishing....8
Figure 2: Image WEB dashboard....9
Figure 3: Main use case TWIST accident publishing....10
Figure 4: Block diagram TWIST real time collection....12
Figure 5: Image TWIST real time collection....13
Figure 6: Main use case TWIST real time collection....14
Figure 7: Image registration process....16
Figure 8: Image login process....16
Figure 9: Customization process....17
Figure 10: Mobile App Figure 11: Mobile App SaveMe....18
Figure 12: Block diagram Air Quality....19
Figure 13: Block diagram Pollution monitor for outdoor activities....27
Figure 14: Image Mobile App Barcelona corre....29
Figure 15: Use Case: Monitoring (real time) pollution in spaces for outdoor activities....31
Figure 16: Block diagram Correlation between environment and mobility....33
Figure 17: Screenshot Correlation between environment and mobility....35
Figure 18: Use caseCorrelation between environment and mobility" 36
Figure 19: Block diagram of road incidents and accessibility....39
Figure 20: Screenshot Mobile App Road incidents and accessibility....41
Figure 21: Use case Road incidents and accessibility....43
Figure 22: Block diagram Data publication....45
Figure 23: Screenshot Data Publication....47
Figure 24: Use case Data publication....49
Figure 25: Block diagram City Dynamics....51
Figure 26: City Dynamics Workflow-I....52
Figure 27: Screenshot Sample request....53
Figure 28: Screenshot result....54
Figure 29: City Dynamics Workflow-II....55
Figure 30: Screenshot list)....56
Figure 31: Screenshot information....57
Figure 32: Screenshot getInformation....58
Figure 33: Screenshot query....59
Figure 34: Mobile App image....60
Figure 35: Mobile App image....61
Figure 36: City Dynamics Deployment Diagram....62
Figure 37: City Dynamics Use-Case1: Get Garbage Collection Hours By Location....63
Figure 38: City Dynamics Use-Case2: Get Garbage Collection Hours....63
Figure 39: City Dynamics Use-Case3: 3rd Party Development....64

Figure 40: City Dynamics Use-Case4: System Administration 65
Figure 41: Block Diagram POIs 67
Figure 42 POI Workflow 68
Figure 43: Screenshot sample request 69
Figure 44: Screenshot request POIs 70
Figure 45: Screenshot request ClosestPOI 71
Figure 46: Screenshot list cities 72
Figure 47: Screenshot request detailed information 73
Figure 48: Screenshot Mobile App POIs 74
Figure 49: Screenshot Mobile App POIs (2) 75
Figure 50: POI Deployment Diagram 76
Figure 51: POI Use Case1: Citizen/Tourist POI Information 77
Figure 52: POI Use Case2: 3rd Party Development 78
Figure 53: POI Use Case3: System Administration 78
Figure 54: Block Diagram Reports 80
Figure 55: Example App (1) 81
Figure 56: Example App (2) 82

1 Overview

This document includes the development reports of the pilots, showing the development results, indicating the resources repositories and providing the way to deploy all pilot applications.

2 Pilots implementation report

All development reports are explained by pilot areas.

2.1 Piedmont Region Pilots implementation

2.1.1 TWIST accident publishing

Abstract	
This pilot planned to deliver API in geographic and tabular data to citizen on a merged data set of accident and traffic. Since the change of access right on the API and in the effort to convince PA stakeholders on the utility of having the two data sets together the project limited the API generation and delivered a dashboard to the Piedmont Region users only that show the accident point and the traffic correlated information.	
Technologies used on pilot	**Data-sources silos**
• Open-DAI platform • Web application based on OpenLayer	• Accident data set • Traffic data set

The pilot is reserved to the Piedmont Region users only. The reason for this user restriction is that both data sets are not considered open by the owner therefore the application has been deployed as reserved to a limited number of users authorized to use the application.

The traffic data set has been obtained thanks to the Open-DAI project for usage for the Public Administration, since is managed and owned by a third party, public owned, company that did not opened it until the Open-DAI project requested.
The traffic data has been given just for internal PA usage and not open to public.
The original plan to open that data as part of the project had to be changed and the usage of that information has been limited to this application for now, but will be used also for other internal development.
In this approach we can consider this as a minor success since this data is now available as a new data set for other project and there is an API that exposes it.
This has limited the initial scope of the pilot that planned to open API to citizen and public usage.

The project thus opted for implementing a simple demonstrator of how these API could be used to give more insight to PA managers with the correlated data.

2.1.1.1 Pilot development

The following diagram shows the deployment of the artefacts of the pilot:
The orange nodes are the Open-DAI platform nodes, while the green ones are the external nodes.
The legacy DB is accessed by geoserver that generate the needed layers. The access to geoserver is mediated by the API Manager that protects the API through ACL allowing only defined users to access the geoserver service.
The Web application is published through an Apache that is exposed only to the internal network of Piedmont Region. The when application uses OpenLayer to query the geoserver data.

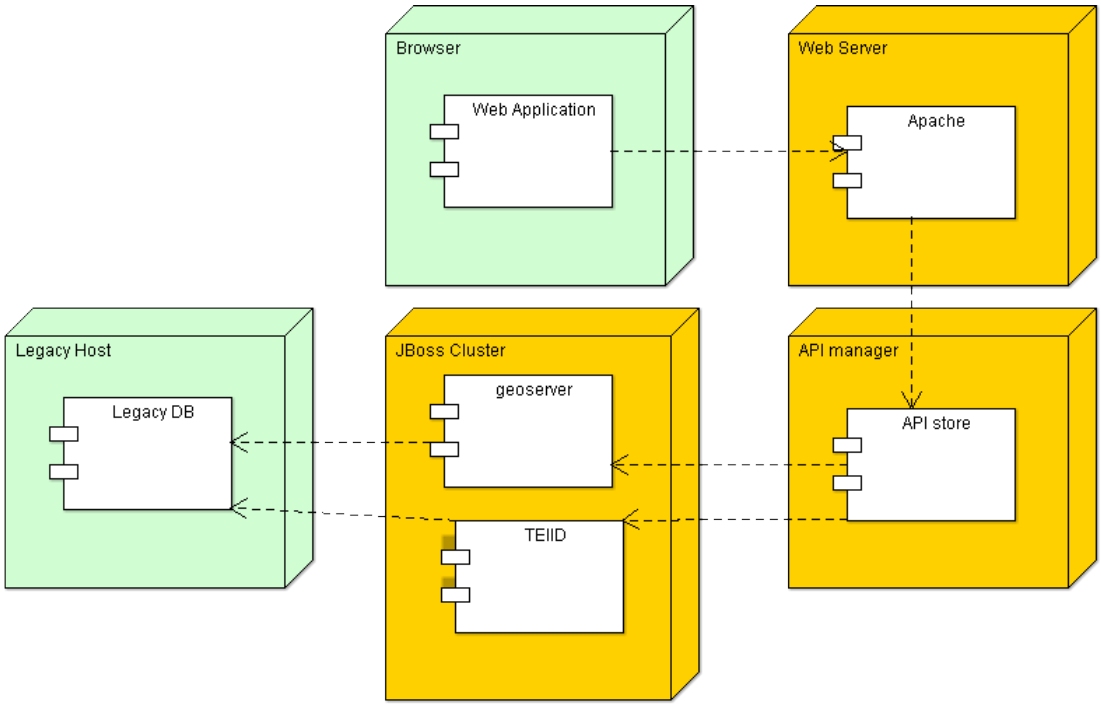
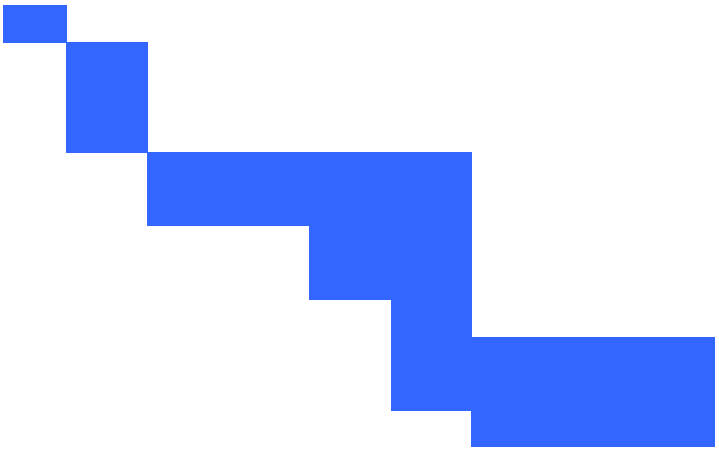


Figure 1: Block diagram TWIST accident publishing

2.1.1.1.1 Followed development plan

	M9	M10	M11	M12	M13	M14	M15	M16	M18
Definition of the traffic data collection procedure									
Definition of the new set of									

accident data
Definition of the correct association between traffic and accident
Implementation of the raw data traffic collection
Implementation of the merged dataset
API publishing
Implementation of web dashboard
Functional and load testing



2.1.1.1.2 Development results

The WEB dashboard is published in the internal intranet of Piedmont Region, while the API is published in the Open-DAI API manager but protected by ACL control and not visible or usable to generic public.

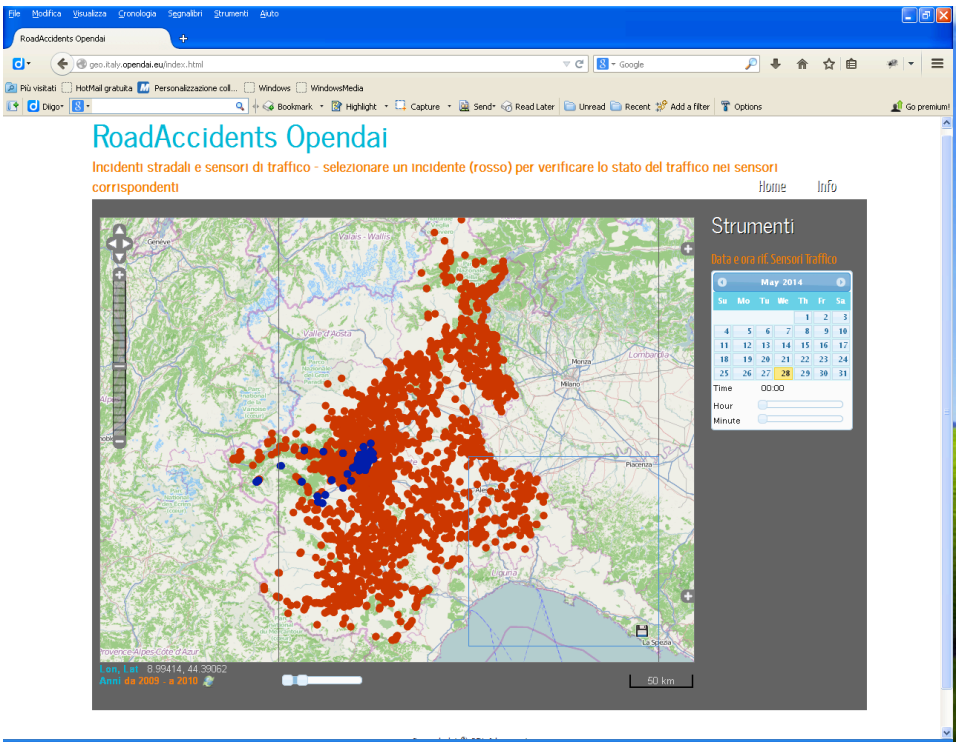


Figure 2: Image WEB dashboard

2.1.1.2 Pilot documentation:

2.1.1.2.1 Pilot code repository

Github repository: https://github.com/open-dai/piedmont-pilot-accident

2.1.1.2.2 How-to install and deploy

The web application has been deployed in an Apache with PHP and it is defined a virtual host dedicated to the web site.

The geoserver services are deployed by copying the workspace information in the corresponding data directory of geoserver. A restart of the geoserver instance it is required.

2.1.1.2.3 User Manual

The main use cases for the web dashboard are represented in the diagram below.

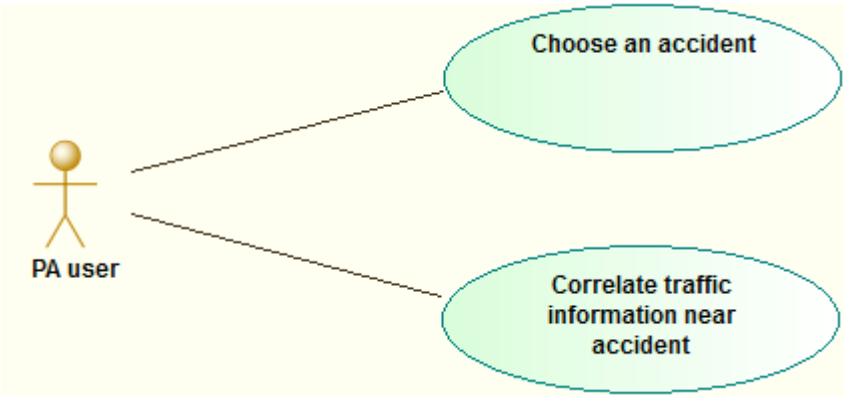


Figure 3: Main use case TWIST accident publishing

The user has a map where all the accident point are plotted.
He can filter by period to get just the accident he is interested.
The map also shows the traffic sensors position.

When he selects an accident he can visualize in the sensor point near the traffic information to correlate the traffic information to the accident event.

The application is javascript based application, mainly based on OpenLayer.

2.1.2 TWIST real time accident collection/SaveUS

Abstract	
Application to collect real time accident data from citizens. Pilot is composed by a web site, a smartphone application, a business process and API	
Technologies used on pilot	**Technologies used on pilot**
• Open-DAI platform • YII framework	• Open-DAI platform • YII framework

The pilot deployed in September 2013 has been replaced by a new version forked by a private company that used the published source to release a new improved and integrated with the emergency service IT systems in the end of 2013.
Since the emergency department is a public actor financed by Piedmont Region, it has been decided to retire the Open-DAI pilot in favour of this new release to make citizens converge on one single solution.

2.1.2.1 Pilot development

The following diagram shows the deployment of the artefacts and the usage of the Open-DAI platform.
In orange the nodes of the Open-DAI platform in green the external nodes.
The web site is the place where end users register to the service and configure the process.

The process is the artefact to be deployed in WSO2 BPS that executes the operations in consequence of the smart phone application usage.

The smart phone application is the end user application that allows to signal the accident.
The internal services are the ones that insert data into the DB and expose the process information to the BPS; these are internal services and not produce any API.

This pilot makes an extensive usage of the Open-DAI components since it involves the adoption of the Business Process Server tool to orchestrate different internal services and to offer to the pilot a single new API to manage the communication of the accident.

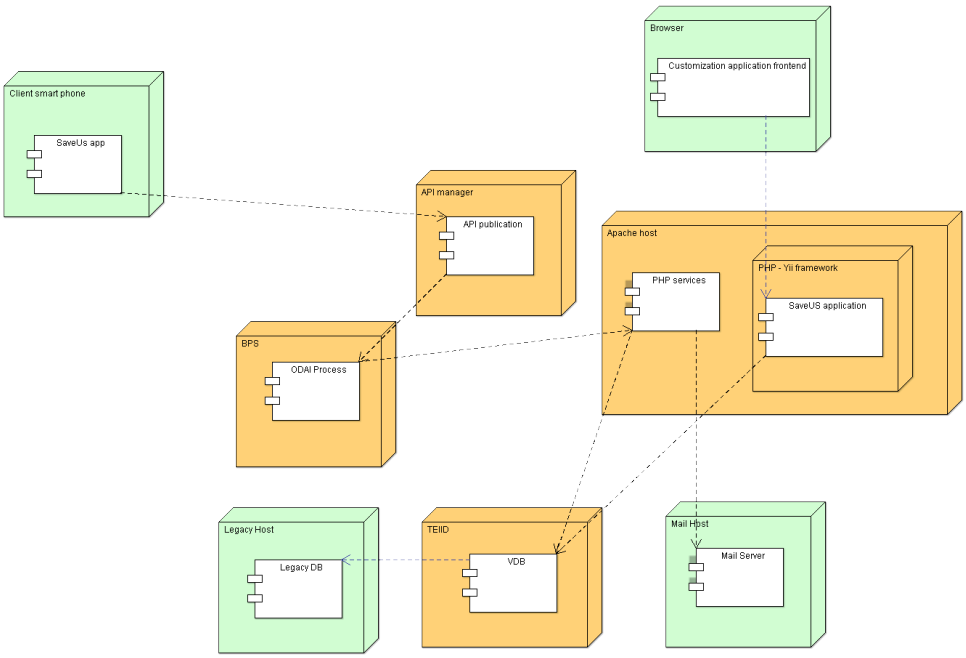
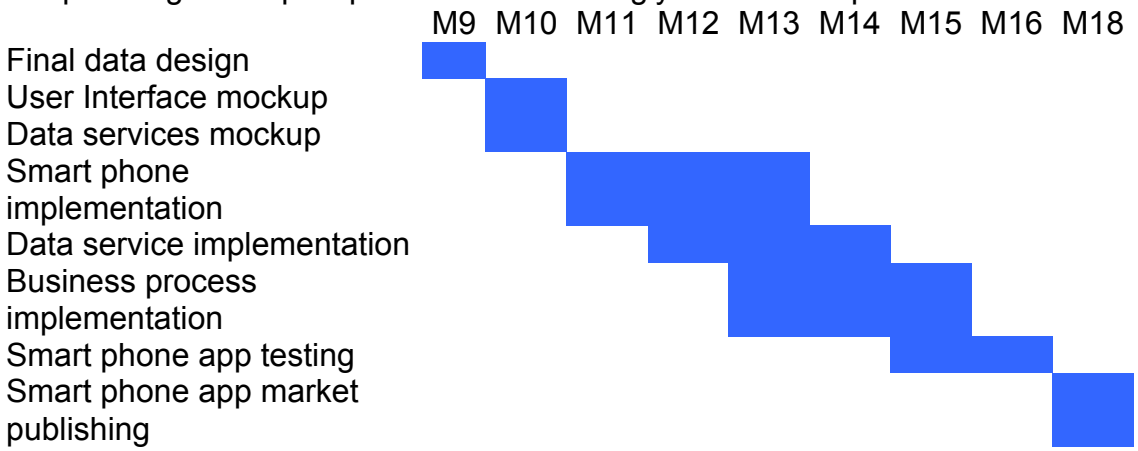


Figure 4: Block diagram TWIST real time collection

2.1.2.1.1 Followed development plan

The planning of the pilot proceeded accordingly to the initial plan


M9 M10 M11 M12 M13 M14 M15 M16 M18
Final data design
User Interface mockup
Data services mockup
Smart phone implementation
Data service implementation
Business process implementation
Smart phone app testing
Smart phone app market publishing

2.1.2.1.2 Development results

The web site is published at the address:
http://saveus.opendai.eu
The API is published by the API manager, but is reserved to the smart phone application and visible just to the appropriate user.

The smart phone application is published in the Apple Store and Google Market.

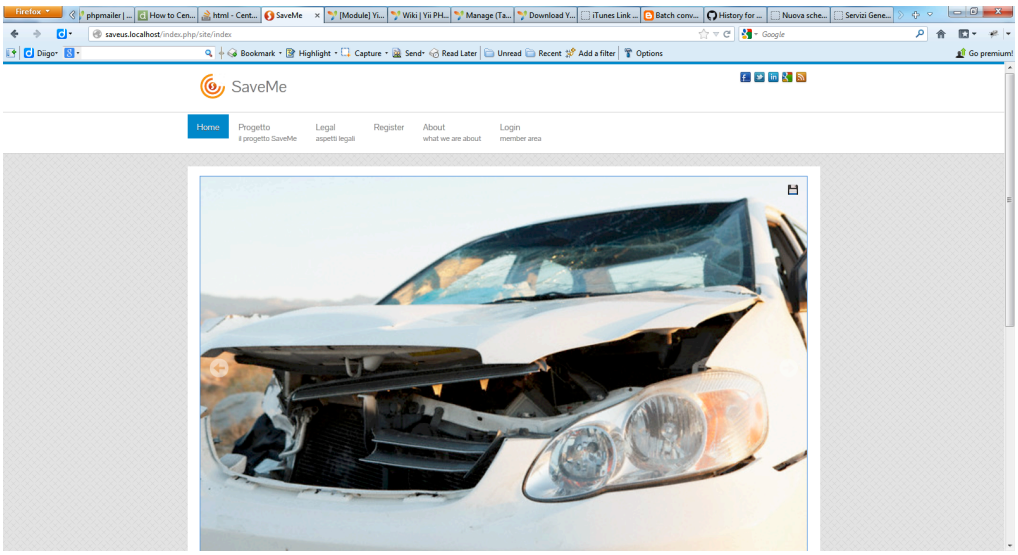


Figure 5: Image TWIST real time collection

2.1.2.2 Pilot documentation:

2.1.2.2.1 Pilot code repository

Github repository: https://github.com/open-dai/piedmont-pilot-savelives

In the source repository can be found 4 different artefacts:

- The save us website
- The smartphone apps (divided in iOS and Android codebase)
- The internal services
- The business process

2.1.2.2.2 Pilot code guide

The following diagram summarize the principal use cases of the pilot:

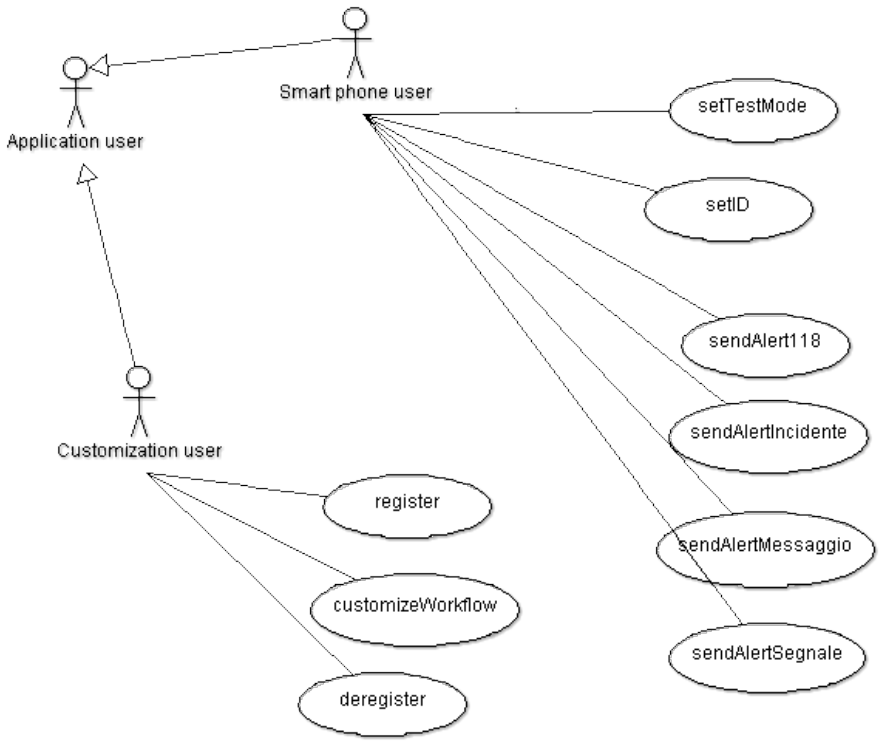


Figure 6: Main use case TWIST real time collection

The user acts on the two different systems with different capabilities.
On the web component he can register and deregister, and can customize the process workflow.
On the smart phone he can configure it and send alerts.

2.1.2.2.3 How-to install and deploy

The web site has to be deployed into an Apache with PHP configured.
A virtual host for the web site has to be created.
The production parameters like production mail server and DB have to be set in the "\webapp\protected\config\main.php" file.

The internal services can be deployed on the same Apache, but there is no need to expose the service to the public since it is used just by the BPS.

The WSO2 BPS archive has to be deployed using the WSO2 deploy console into the BPS server.

The smart phone application have to be compiled and signed in order to be deployed in the Apple and Google marketplace.

The TEIID component virtualizes the new dataset and is used both by the YII web application and by the PHP services.
The web application manages the customization data, while the PHP services offer API used by the BPS to get the information.
Just the API from the BPS is exposed through the API manager and is reserved to the smart phone application; the API manager is enforcing protection on this API through ACL.
The PHP services are also used to send mails from the BPS process.

2.1.2.2.4 User Manual

There are just 2 element of the pilot that require a “user manual” since all the other components are just server side components.
Web Site
The web site offers the following main features

- Registration
- Login
- Process customization

These use cases will be hereafter described

Registration

The user accesses the registration form of the web site and inserts the data.
As a consequence he will receive a confirmation eMail with an activation link and the code to insert in the smart phone application; this code will be the only element that will be used to allow the BPS to get the process customization; this way no user data transit in the systems and since this code is not saved with the accident information we can assure that is not possible to associate the accident information to a particular user preserving anonymity.

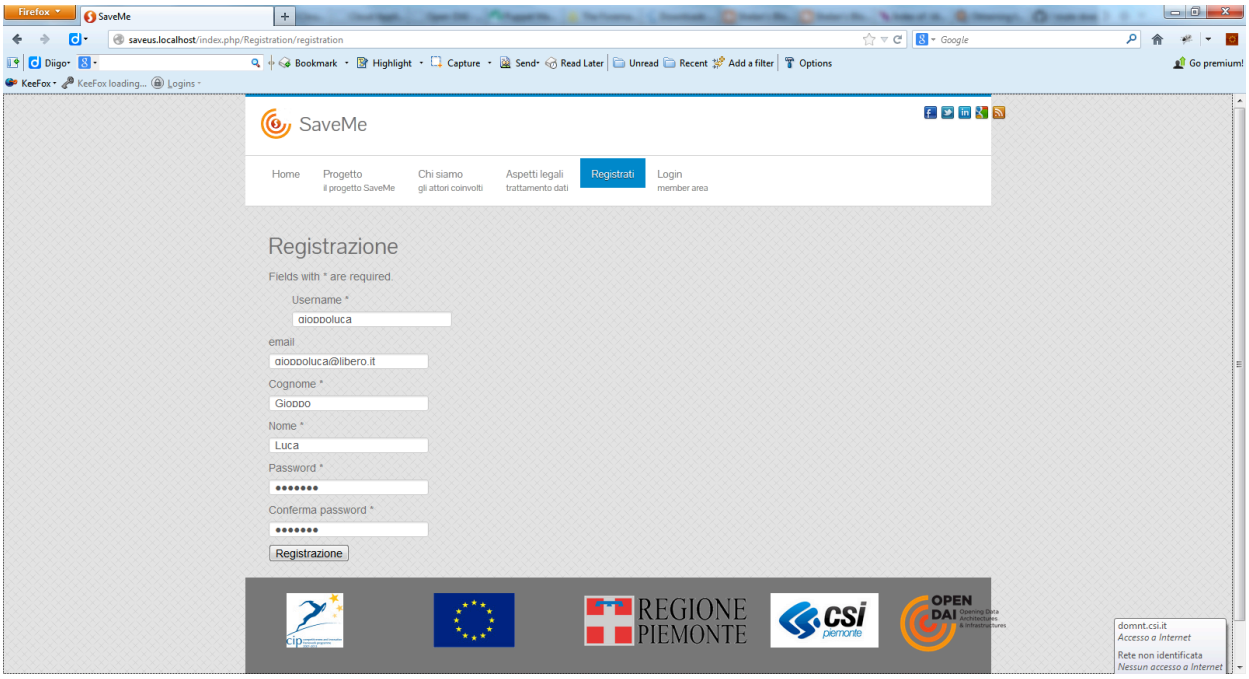


Figure 7: Image registration process

Login

With the login data it will be possible to login into the web site and doing so will enable a new tag for managing the process customization.

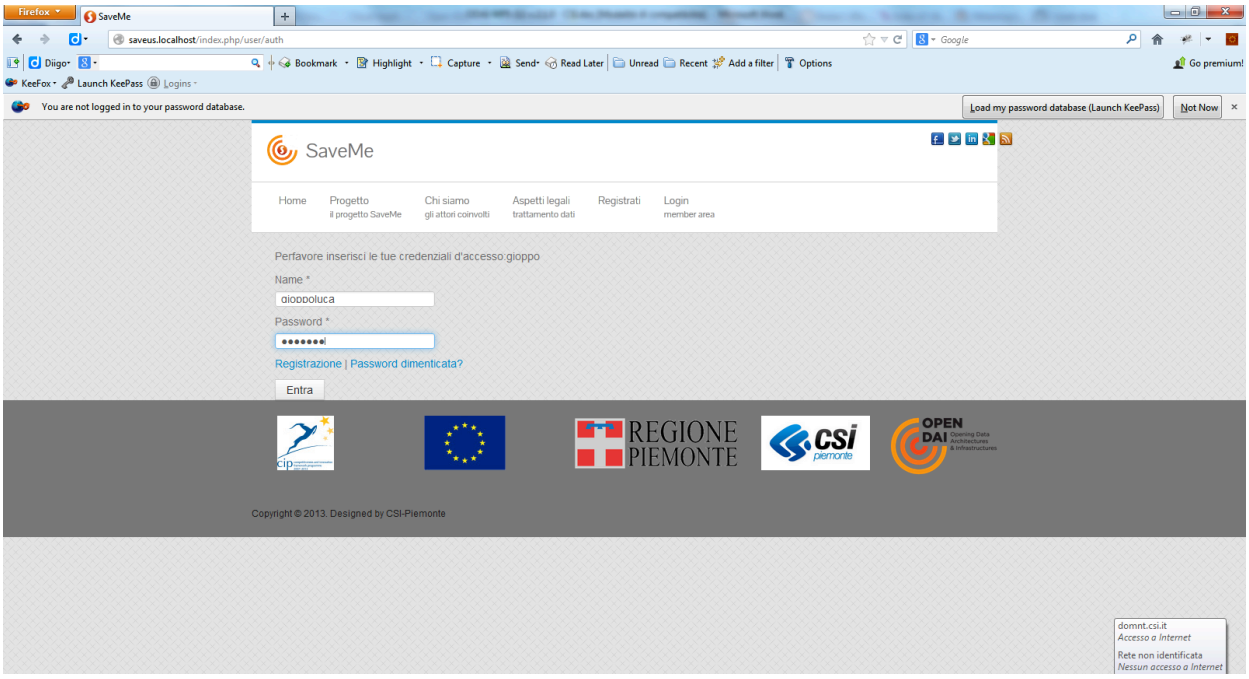


Figure 8: Image login process

Process customization

The process is saved and managed as an XML file to allow for extensibility in future evolution of the system. The whole XML data is presented to the user as a sequence of tabs and accordion elements. At the moment the only operation allowed is sending a mail, but in the future it will be possible to implement new operations.
The user has now the option of entering a mail address and a message that will be sent by the BPS through the internal services.

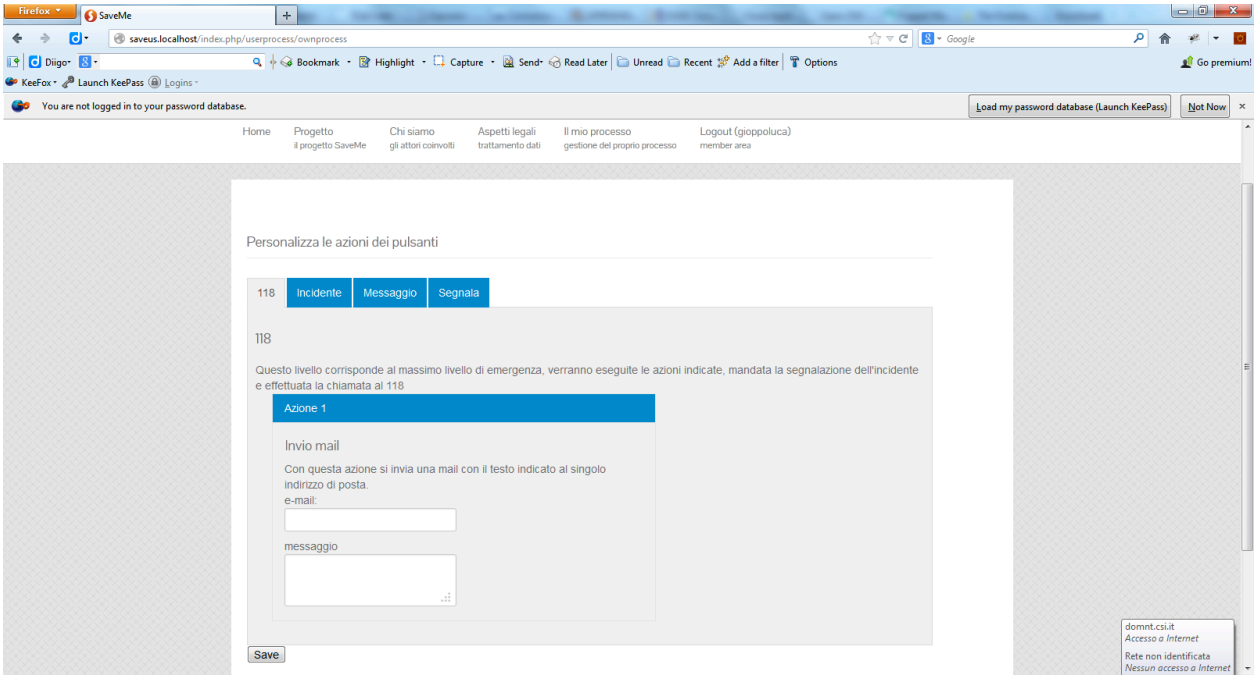


Figure 9: Customization process

SmartPhone app

The smart phone application is as simple as possible.
There is a settings area where the user will enter the code received in the registration mail.
The user can also set the application in testing mode to prevent the application to save the data or calling the emergency number, but testing the mail sending to assert that the application is really working.
Setting this option activate a blinking message in the main screen.

Figure 10: Mobile App

Figure 11: Mobile App SaveMe

The main screen offers just the 4 buttons that allows to send the message and signal the different types of emergency as described in the pilot description.

2.1.3 Air Quality

Abstract	
API for publishing air quality information accordingly to EU standard	
Technologies used on pilot	**Data-sources silos**
• Open-DAI platform • Java	• Air quality dataset

2.1.3.1 Pilot development

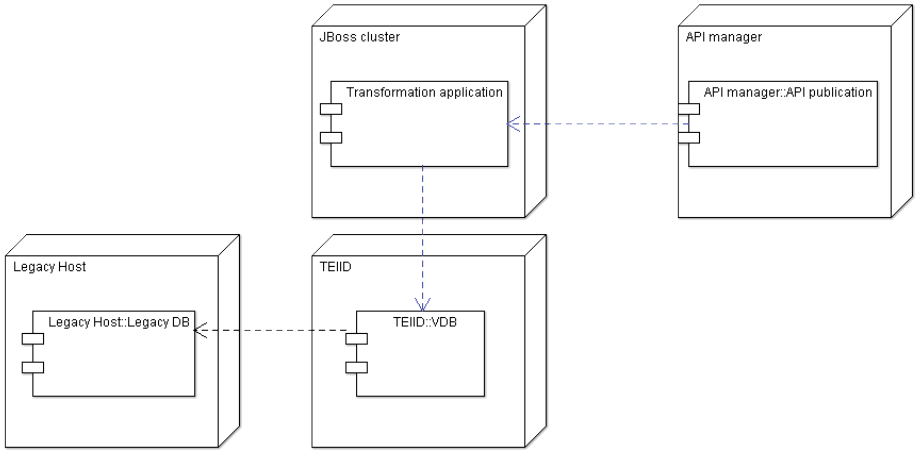


Figure 12: Block diagram Air Quality

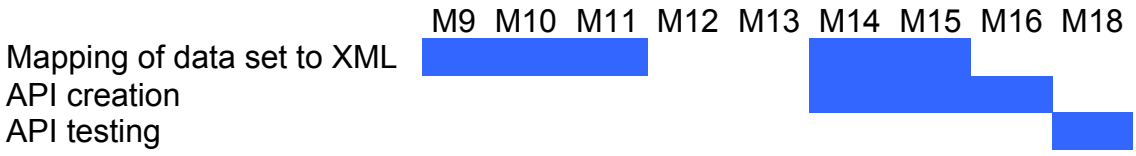
The diagram above shows the deployment diagram of the artefacts: the TEIID VDB accesses the legacy DB; the java application connects to the VDB and the API thus generated is published through the API manager.

The project made a first attempt in using just TEIID to expose the API by transforming the relational view into the XML output needed.
The complexity of the EU adopted schema and the fact that many xschema referenced URI not valid made impossible to use the wizards in the TEIID designer and the process over complicated.
The development thus decided to create a set of java classes to marshal and unmarshal the content and from these to build the objects needed to execute the mapping for the correct output.

Since the Open-DAI architecture provisioned a JBoss cluster it was possible to solve the initial problem (that consumed part of the initial activity) without diverging from the project specifics.
The development has been done in synch with the meetings with the European Environment Agency where CSI and Piedmont Region participate to meetings for the standard definition.

The creation of the mapping of the data and API took longer than planned due to the complexity of the XML document to be produced.

2.1.3.1.1 Followed development plan



2.1.3.1.2 Development results

The API can be found at
https://api.opendai.eu/store/apis/info?name=AirQuality&version=1.0.0&provider=ItalyAdmin

2.1.3.2 Pilot documentation:

2.1.3.2.1 Pilot code repository

Github repository: https://github.com/open-dai/piedmont-pilot-air

In the source repository there is the eclipse project to create the java servlet that transforms the original data into the EU standard data format, since the required XML was too complex to be managed by the TEIID translator.

2.1.3.2.2 Pilot code guide

The request is done by the API consumer to the API manager that turns the requests to the JAVA servlet that invoke the DB to get the information, than thanks to the XML mappings creates the correct XML response from relational data.
The mapping from the relational and the XML schema is done using XMLBeans that is a tool that generates JAVA beans compliant to an XML schema so it will be possible to populate the JAVA beans and marshal the resulting XML easily.

2.1.3.2.3 How-to install and deploy

The code produces a WAR file that has to be installed in a JBoss application server.
The application server requires the JNDI connection to the DB and the servlet exposes the data service that will be published through the API manager.

2.1.3.2.4 User Manual

This pilot is a pure API project.
There are 4 API published to expose the different entities that are:

- Measurement network stations
- Measurement stations
- Measurement parameters
- Raw data measurement

The data is real time data.
The API call is described in the following table

API	result	parameter	parameter meaning
/networks	List of measurement networks		
/networks/{id}?startDate=AAAMMMGGG&$endDate=AAAMMMGGG	List of stations part of the network	Id	Id of the network
		startDate	Optional date to require stations active from date
		endDate	Optional date to require stations active until date
/stations	List of all stations	startDate	Optional date to require stations active from date
		endDate	Optional date to require stations active until date

/stations/{id}?startDate=AAAMMMGGG&$endDate=AAAMMMGGG	List of sensors of a station	Id	Id of the station
		startDate	Optional date to require stations active from date
		endDate	Optional date to require stations active until date
/parameters	List of parameters		
/data?stations=(idstation1:idpar1,idpar2;idstation2:idpar3)& startDate=AAAMMMGGG& $endDate=AAAMMMGGG	The measured data for required parameters of the stated stations	stations	The list of sensors of each station to get data
		startDate	Optional date to require data from date
		endDate	Optional date to require data until date

The documentation on the entity returned by the API can be found on the EU web site.
http://dd.eionet.europa.eu/schemaset/id2011850eu/view
http://dd.eionet.europa.eu/schema/id2011850eu/20130712_IPR_Mapping.xlsx/view

2.1.4 Transport API

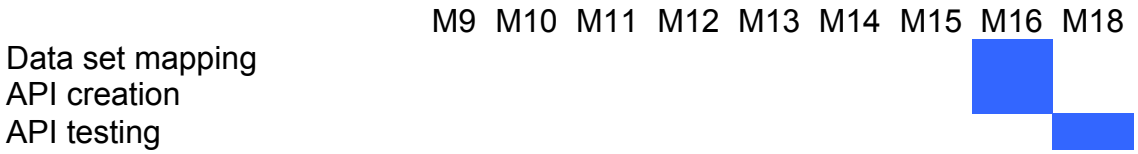
Abstract	
API publishing transports information	
Technologies used on pilot	**Data-sources silos**
• Open-DAI platform	• Transport dataset

2.1.4.1 Pilot development

This pilot suffered by a DB migration problem due to the fact that the legacy owner started a migration towards a new software system after the project start.
This caused a reduction in the scope of the project and a reduction of the data set that has been published. Development had to delay until the first data set was migrated and was stable enough to implement API on it.
The adoption of a VDB helped the project to identify a small subset of information and defining a very simple view model for data so that it was possible to adapt the virtual view to the changing structure of the legacy DB.
The project team chose the JSONP format for opening to HTML and javascript consumption of the API solving the cross-scripting problem.

This pilot suffered from a delay in the development due to the change of the data set stated above.

2.1.4.1.1 Followed development plan



2.1.4.1.2 Development results

The API can be found at:
https://api.opendai.eu/store/apis/info?name=tpl_movicentri&version=1.0.0&provider=ItalyAdmin
https://api.opendai.eu/store/apis/info?name=tpl_aziende&version=1.0.0&provider=ItalyAdmin

2.1.4.2 Pilot documentation:

2.1.4.2.1 Pilot code repository

Github repository: https://github.com/open-dai/piedmont-pilot-tpl

In the source repository there is the TEIID project to create the VDB and the PHP scripts that implement the API.
The PHP scripts connect to the VDB through the postgres protocol and the Slim framework implement the REST API that are published by the API Manager

2.1.4.2.2 How-to install and deploy

There are two artefacts for this project:

- The VDB
- The PHP rest service

The VDB has to be deployed in TEIID accordingly to the Open-DAI platform.
The PHP rest service has to be published in the internal Apache server.
The API is than published by the API manager.

2.1.4.2.3 User Manual

The API is divided into two main API one dedicated to the “aziende” entity. This entity represents the transport companies that operate in the Piedmont region.
The second API is dedicated to the “Movicentri” entity that represents the nodal point of stop and departure of different transportation systems

Tpl_aziende

API	result	parameter	parameter meaning
	List of aziende	NA	No parameter required

Tpl_movicentri

API	result	parameter	parameter meaning
/	Information on transport event in the movicentro	_callName	Type of call required “**getSegnalazioni**”
		movicentro	The ID of the movicentro

/	Get the list of the movicentri around a point	_callName	Type of call required "**getMovicentri**"
		longitude	The point from where you make the search
		latitude	The point from where you make the search
/	Get the Arrivals time in a movicentro	_callName	Type of call required "**getArrivi**"
		movicentro	The ID of the movicentro
/	Get the Departures time in a movicentro	_callName	Type of call required "**getPartenze**"
		movicentro	The ID of the movicentro

2.2 Barcelona municipality Pilots

2.2.1 Pollution monitor for outdoor activities Pilot (Mobile App)

Abstract	
The pollution monitor for outdoor activities pilot is a mobile application with relevant data about the environmental conditions on the Barcelona area, in order to choose the best places or routes for outdoor activities or sports. This pilot also represents opening the access to the environmental data for Barcelona.	
Technologies used on pilot	**Data-sources silos**
• Open-DAI platform • REST/JSON • Python • Django framework • Titanium framework	• City facilities • Street directory • Pollution levels • Noise Levels • Pollen levels • Weather

This pilot aims to publish the data silos detailed on the document D5.1 using the Open-DAI infrastructure and technologies. Once APIs are available the pilot aims to develop a specific utility to use this data and prove the use case detailed also on D5.1.

The Pilot development consists on adding APIs on Open-DAI platform from silos previously specified and development of a mobile application addressed to the Barcelona citizens. The main goal is to achieve a useful service using the APIs opened with Open-DAI project.

The service developed to prove that the data is useful is a mobile app that provides environmental data on real-time to the runners. This App was not developed from scratch. This pilot is an extension on existing mobile application from Barcelona city council that contains the running routes on the city. This App had already a good acceptance on the city but with Open-DAI project we have added dynamic and real time data improving significantly the app features and the rating among the users.

2.2.1.1 Pilot development

In order to develop the application “Monitoring pollution in spaces for outdoor activities” in a service, we define a solution composed by a Mobile Application as a user interface and a service back-end to attend the petitions from the mobile phone.

The smart phone application interacts with the server application described and offers a graphical user interface for the consumers. This mobile application is developed using a web based cross-platform technology in order to facilitate the availability on the different mobile platforms.

The following diagram shows the deployment of the artefacts of the pilot:

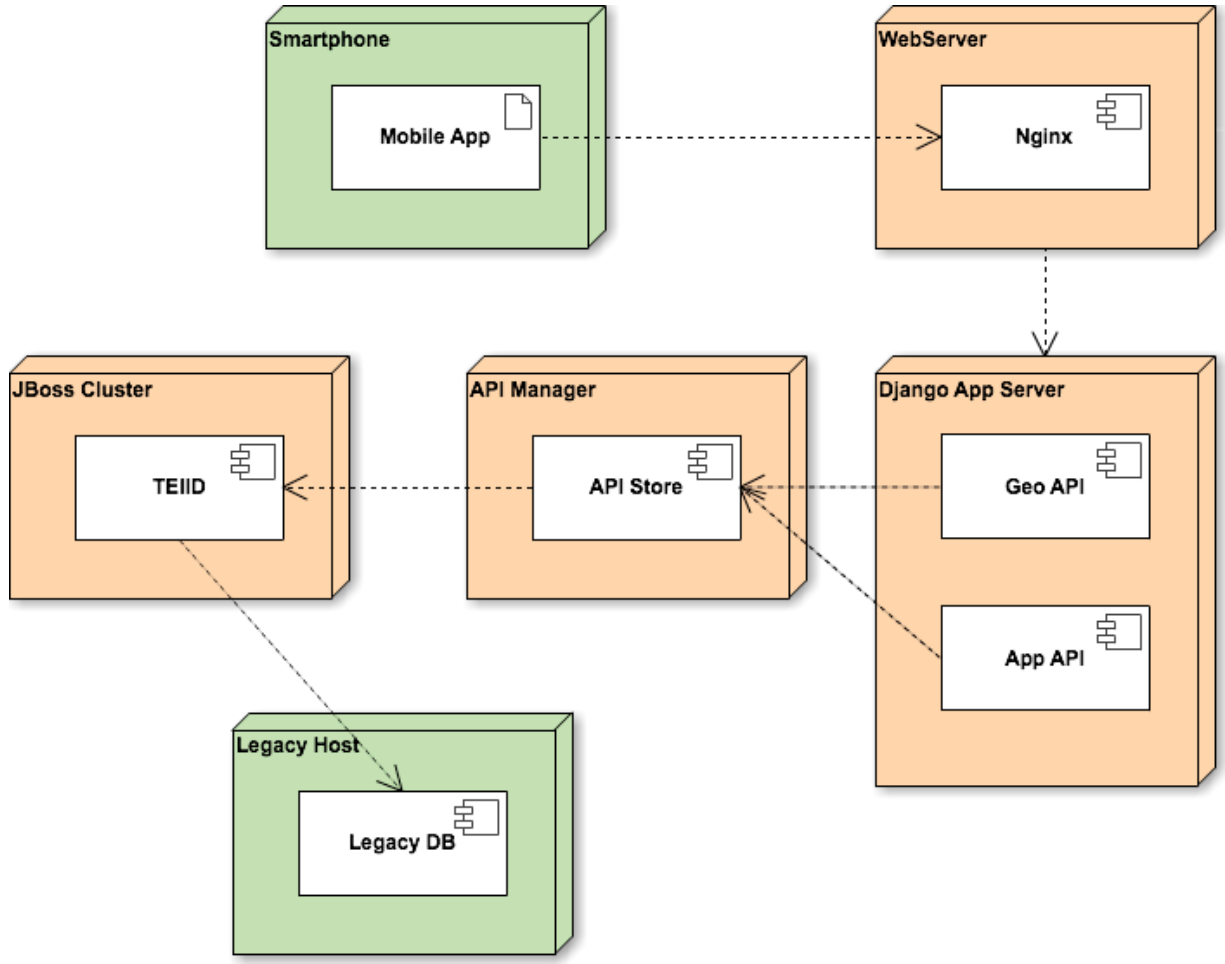


Figure 13: Block diagram Pollution monitor for outdoor activities

In the diagram there are two types of artefacts. The orange nodes are the Open-DAI platform nodes, while the green ones are the external nodes.

Describing the artefacts on detail:

- On Public administration host are localized the legacy data sources connected to the Open-DAI Platform.
- The first artefact as part of Open-DAI platform is the TEIID data virtualization system adapted to Barcelona city silos.

- Over the data virtualization the API manager provided by WSO2 framework offers access and control to the REST APIs publishing publicly an API store.
- Also part of Open-DAI platform is designed and developed using Django framework the back-end required by the mobile application. This back-end is composed by 2 artefacts:
  - A Geo-server artefact developed using geodjango required by mobile app for the geospatial services such as geonaming reverse resolution and some different trigonometric functions.
  - An application server developed using django with services and logic functions required by mobile application on purpose to light the computation on the mobile device.
- The access on the back-end services is provided and controlled by a proxy implemented with NGinx.
- At the end a cross platform mobile application is developed using Titanium framework as a client of the services and as a user interface application.

2.2.1.1.1 Followed development plan

M9 M10 M11 M12 M13 M14 M15 M16 M18
Pollution dataset connection procedure
Weather dataset connection procedure
Noise dataset connection procedure
Pollen dataset connection procedure
Development TEIID data virtualization
App Backend development
App Mobile development
App testing
Deploying and publishing

2.2.1.1.2 Development results

The pilot results are mainly 2 different services, The APIs with public data available for citizens and 3rd party developers and a mobile application focused on adding a service on Barcelona city council public catalogue of services for the citizens.

APIS:

The APIs developed and published on this pilot are:

- Pollution API:

https://api.opendai.eu/store/apis/info?name=Barcelona_Pollution&version=1.0&provider=SpainAdmin

- Pollen API: https://api.opendai.eu/store/apis/info?name=Polen_Barcelona&version=1.0&provider=SpainAdmin
- Noise API: https://api.opendai.eu/store/apis/info?name=Barcelona_Noise_Levels&version=1.0&provider=SpainAdmin

These APIs are explained and documented on the deliverable D6.1 as part of WP6.

App:

The result is an operational app application deployed on iOS and Android markets by the Barcelona city council in order to grow the on-line services to the citizens.

http://apps4bcn.cat/app/barcelona-corre-1/354

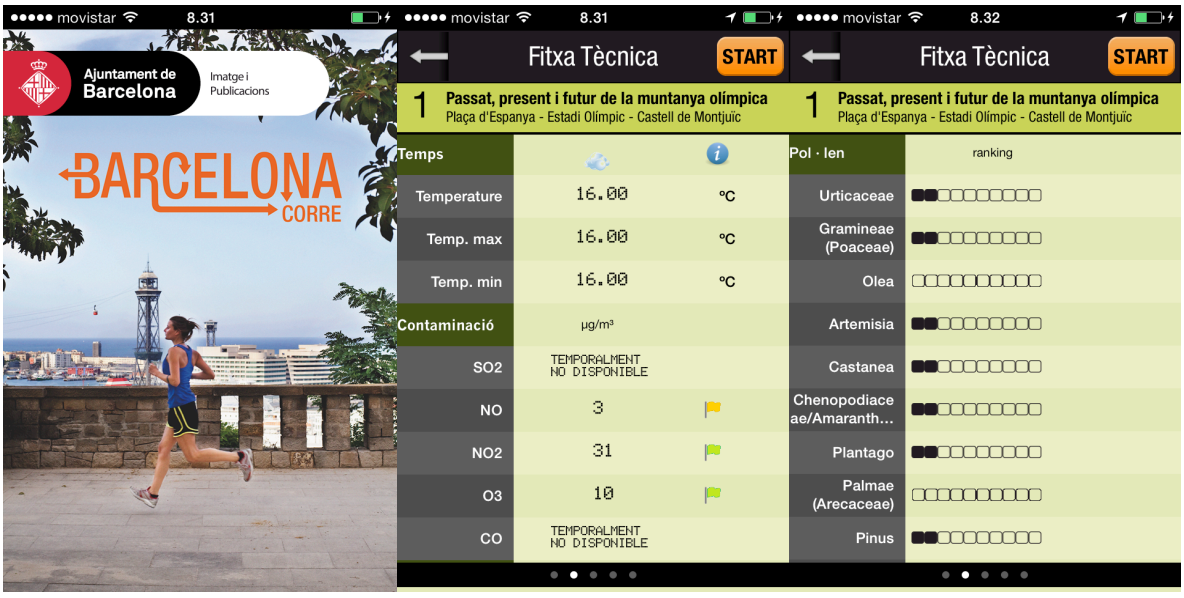


Figure 14: Image Mobile App Barcelona corre

2.2.1.2 Pilot documentation:

2.2.1.2.1 Pilot code repository

Github repository: https://github.com/open-dai/bcn-lleida-opendai-pilots

On Github repository we can found all the code necessary for Pilot divided on

- **jboss-teiid-apis**: the code to implement the TEIID data virtualization to connect to original data-sources (Silos)
- **mobile-apps**: the mobile application code
- **web-geo-server**: the application back-end including the Geo-Server and the

remote services needed by mobile app.

2.2.1.2.2 How-to install and deploy

The process to install this pilot is the same as to deploy the Open-DAI platform, it is detailed on the WP4 deliverables. Inside this platform this Pilot uses:

- JBoss server and TEIID tool for the data virtualization
- Virtual machines for the application servers
- API Manager for Api publication

As it has been previously explained this pilot is composed of a mobile App, and a backend service over the Open-DAI infrastructure. To deploy these artefacts step by step we need to follow:

Data virtualization artefact:

The pilot calls to REST API from Open-DAI WSO2 infrastructure. This API is composed by a WAR (apibarcelona.war) for JBOSS and custom Translators for TEIID.

To deploy this API the source code and the JBOSS + TEIID environment are required. The data used by the pilot is virtualized on a virtual DB (barcelona_base_vdb.vdb) deployed also in TEIID.

All configuration required is included on the war file and this should be deployed as a web application on JBOSS server included in this case on Open-DAI platform.

Application Back-end:

The Backend service is developed using Python-django framework. To deploy these services it requires the source code and a server with Python 2.7 and Django 1.6.x

The django applications run using WSGI technology behind a proxy web server. In our case, we used Gunicorn as WSGI provider and NGINX server as a proxy.

This application has to run on the same network where the TEIID artefacts publish the APIs. Then what it is need is to configure the IP address of this APIS on the django application settings.

Application Front-end:

The Mobile App is developed using Titanium framework. This is a cross-platform technology allowing the deployment on different App markets (iOS and Android).

Items required in order to deploy the apps are: the source code, the Titanium Studio Environment 3.1.x and the Apple and Android market environments.

Once the Android and iOS apps are built with the titanium studio is necessary to follow the official methodology to deploy Apps on the Google and Apple markets.

2.2.1.2.3 User Manual

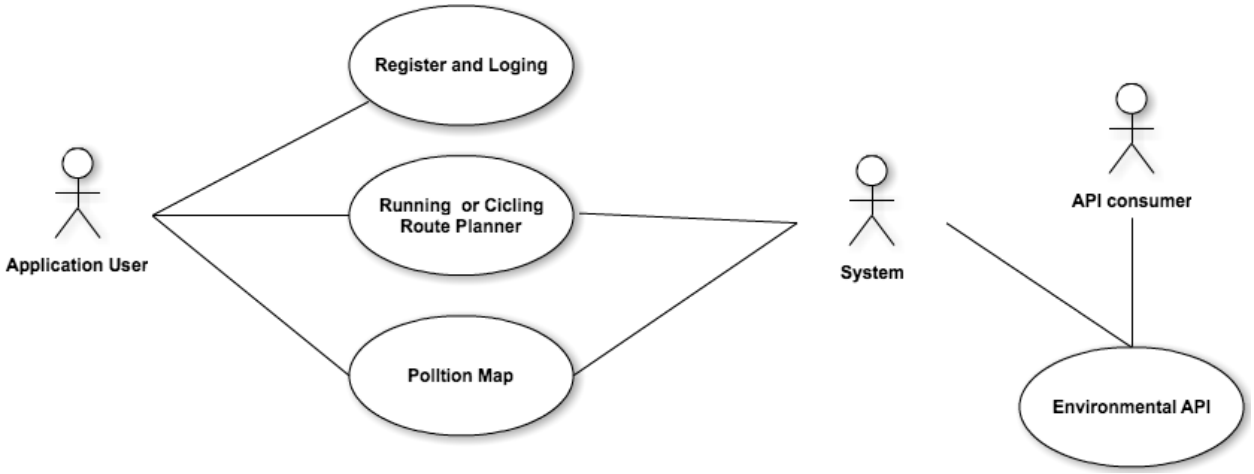


Figure 15: Use Case: Monitoring (real time) pollution in spaces for outdoor activities

This App service has the functionality to show the state of pollution and meteorological factors on different running routes used by citizens. The aim is to provide advice on using the outdoor spaces depending on the levels of pollution or weather conditions. The polluting elements to be taken into account are: The air pollution, the pollen level and the meteorological parameters that can affect the pollution status.

The service logic is responsible to obtain the environmental data from Open-DAI and detect the health risk zones depending on the pollution levels on a specific time. The pollution limits set by the European Union are used as a health risk. Also the service logic takes into account the weather data effect over pollution. Finally, this data service is presented on a mobile application on each running route selected.

2.2.2 Correlation between environment and mobility (Web App)

Abstract	
The Correlation between environment and mobility pilot is a web application with relevant data about the environmental conditions and mobility data from the Barcelona city in order to show on a map view the correlation between the environmental data and the mobility data. This pilot is also opening the silos with mobility data on the city and the access to the environmental data for Barcelona.	
Technologies used on pilot	**Data-sources silos**
• Open-DAI platform • REST/JSON • Python • Django framework • Titanium framework	• Road congestion • Noise Levels • Pollution level

This pilot aims to publish the data silos detailed on the document D5.1 using the Open-DAI infrastructure and technologies. Once APIs are available the pilot aims to develop a specific utility to use this data and prove the use case detailed also on D5.1.

The Pilot development consists in adding APIs on Open-DAI platform from silos detailed and a mobile application development addressed to the Barcelona citizens. One of the goals is to achieve a useful service using the APIs opened with Open-DAI project.

The service developed in order to validate the data utility is a Web application for the public authorities to see the correlation between the air pollution and the traffic status.

2.2.2.1 Pilot development

In order to develop the application Correlation between environment and mobility in a service we define a solution composed by a Web Application as a user interface and a service back-end that attends to the petitions from the mobile phone.

The Web application interacts with the server application described before and offers a graphical user interface for the consumers.

The following diagram shows the deployment of the artefacts of the pilot:

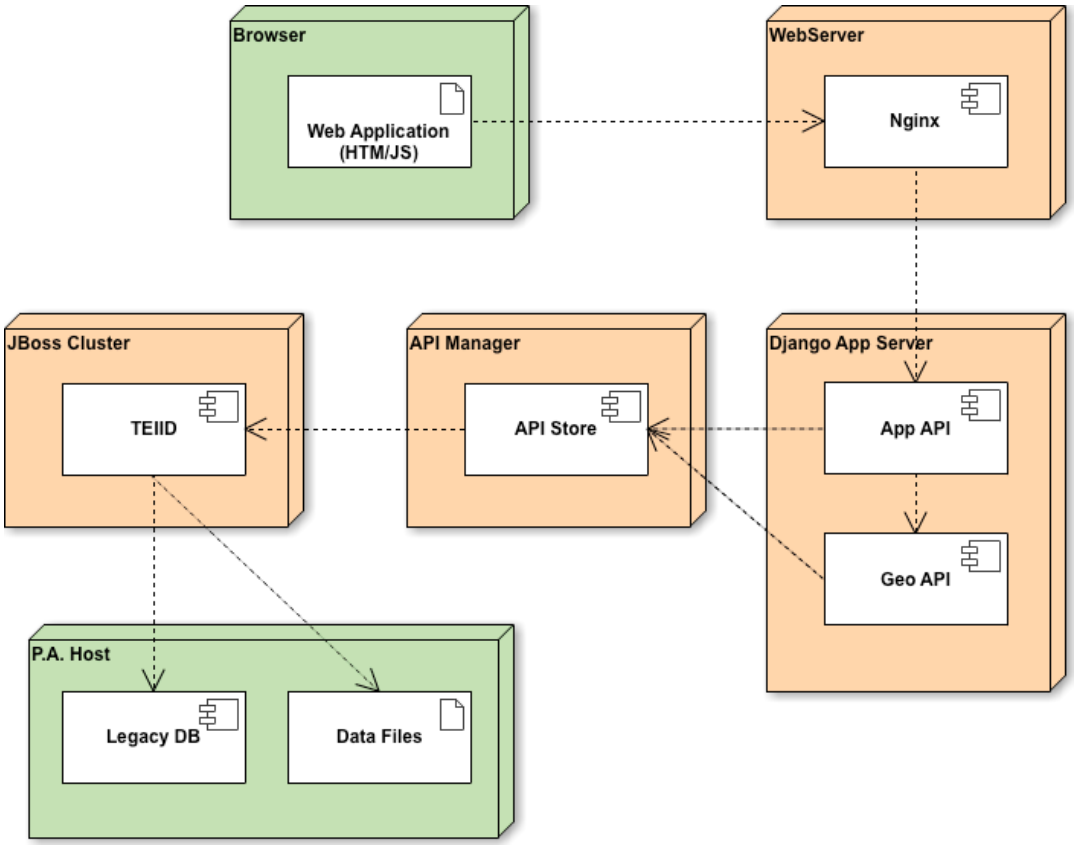


Figure 16: Block diagram Correlation between environment and mobility

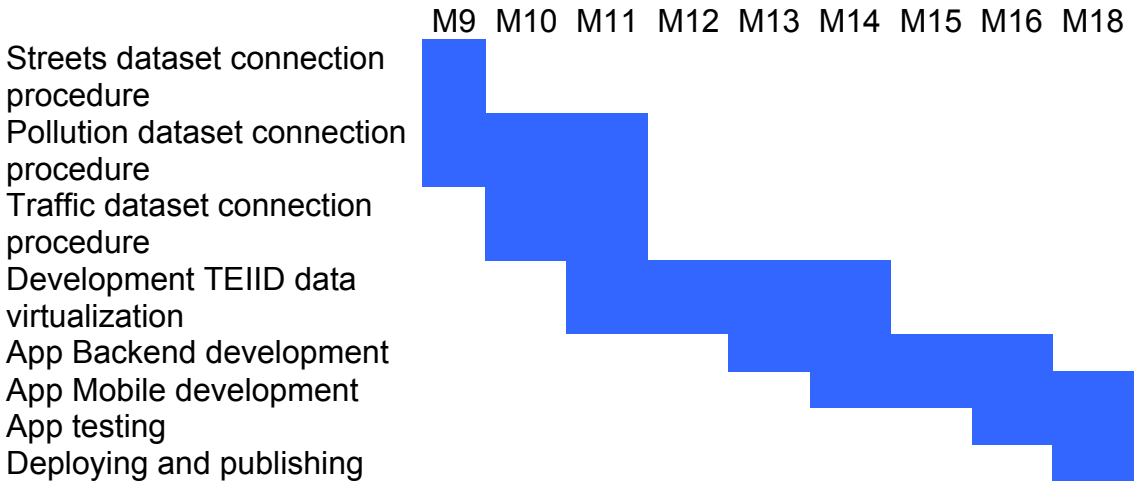
On diagram there are two types of artefacts. The orange nodes are the Open-DAI platform nodes, while the green ones are the external nodes.

Describing the artefacts on detail:

- On Public administration host are localized the legacy data sources connected to the Open-DAI Platform.
- The first artefact as part of Open-DAI platform is the TEIID data virtualization system adapted to Barcelona city silos.
- Over the data virtualization the API manager provided by WSO2 framework offers access and control to the REST APIs publishing publicly a API store.
- Also part of Open-DAI platform is designed and developed using Django framework the back-end required by the mobile application. This back-end is composed by 2 artefacts:
    - A Geo-server artefact developed using geodjango required by the web application for the geospatial services such as geonaming reverse resolution and some different trigonometric functions.
    - An application server developed using django with services and logic functions are required by the web application on purpose to light the computation on the web browser.
- The web resources and the access on the back-end services are provided and controlled by a proxy implemented with NGinx.

- At the end the web application runs on a Web browser providing a user interface for the user.

2.2.2.1.1 Followed development plan



2.2.2.1.2 Development results

The pilot results are mainly 2 different services, The APIs with public data available for citizens and 3rd party developers and a web application focused on adding a service on Barcelona city council catalogue of services for internal public authorities.

APIS:

The APIs developed and published on this pilot are:

- Pollution API: https://api.opendai.eu/store/apis/info?name=Barcelona_Pollution&version=1.0&provider=SpainAdmin
- Streets API: https://api.opendai.eu/store/apis/info?name=Barcelona_Streets&version=1.0&provider=SpainAdmin

This APIs are explained and documented on the deliverable D6.1 as part of WP6.

App:

The result is an operational web app application published on this side:

http://bcn.opendai.eu/

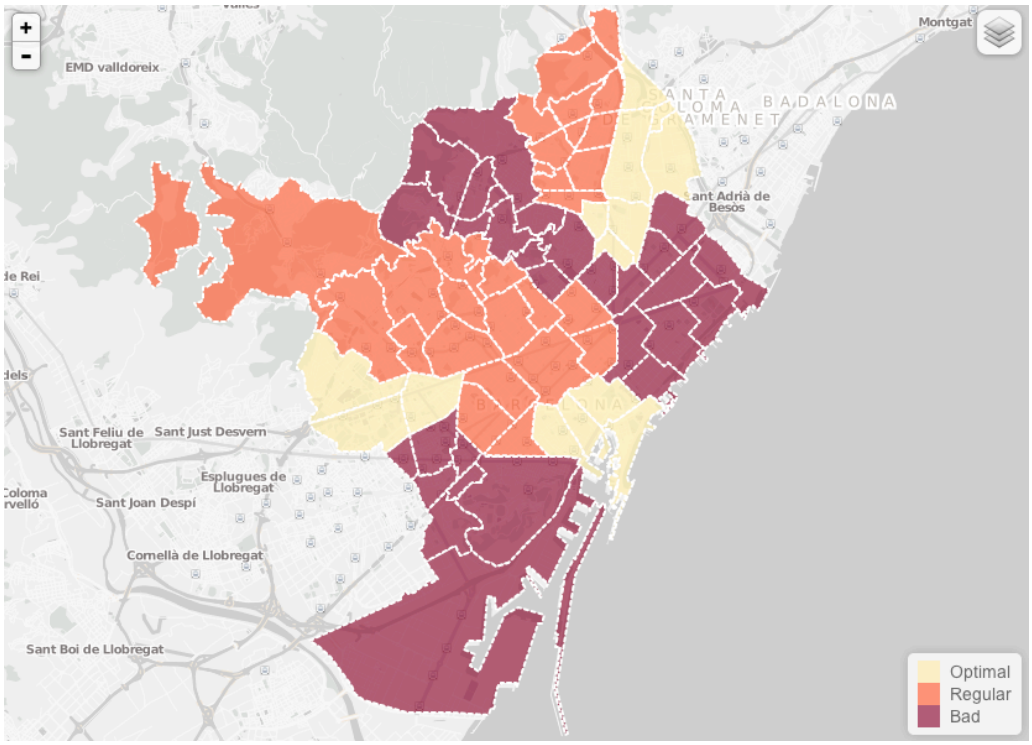


Figure 17: Screenshot Correlation between environment and mobility

2.2.2.2 Pilot documentation:

2.2.2.2.1 Pilot code repository

Github repository: https://github.com/open-dai/bcn-lleida-opendai-pilots

On Github repository we can found all the code necessary for Pilot divided on:

- **jboss-teiid-apis** : the code needed to implement the TEIID data virtualization to connect to original data-sources (Silos)
- **web-geo-server** : the application back-end including the Geo-Server and the resources needed for the web app.

2.2.2.2.2 How-to install and deploy

The process to install this pilot is the same as to deploy the Open-DAI platform, it is detailed on the WP4 deliverables. Inside this platform this Pilot uses:

- JBoss server and TEIID tool for the data virtualization.
- Virtual machines for the application servers.
- API Manager for API publication.

As it has been explained previously this pilot is composed of a mobile app, a backend service over the Open-DAI infrastructure. In order to deploy these artefacts step-by-step we need to follow:

Data virtualization artefact:

The pilot calls to REST API from Open-DAI WSO2 infrastructure. This API is composed by a WAR (apibarcelona.war) for JBOSS and custom Translators for TEIID.

To deploy this API is required the source code and the JBOSS + TEIID environment. The data used by the pilot is virtualized on a virtual DB (barcelona_base_vdb.vdb) deployed also in TEIID.

All configuration required is included on the war file and this should be deployed as a web application on JBOSS server included in this case on Open-DAI platform.

Application Back-end:

The Backend service is developed using Python-django framework. To deploy these services is required the source code and a server with Python 2.7 and Django 1.6.x.

The django applications run using WSGI technology behind a proxy web server. In our case, we used Gunicorn as WSGI provider and NGINX server as a proxy.

This application has to run on the same network where the TEIID artefact publishes the APIs. Then only is needed to configure the IP address of this APIS on the django application settings.

Application Front-end:

The Web App resources needs to be served using a web server system, in our case is used a NGinx software.

2.2.2.2.3 User Manual

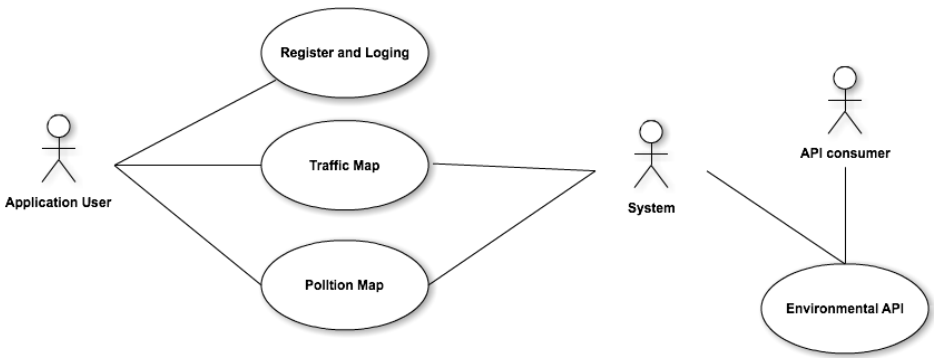


Figure 18: Use caseCorrelation between environment and mobility"

This app permits a data correlation service between environment and mobility data. The main client is intended to be the PA with the goal to improve its planning and operating capabilities. The map displaying services on environmental status will be used in conjunction with urban mobility data obtained by the project. Having environmental status data with location and timestamp attributes will enable to cross this information with other datasets like mobility reports.

These data relationship will allow to public authorities to formulate and modify (e.g. in a daily basis) a mobility plan for the city that takes into account also the pollution factor (i.e. car-generated pollution such as quality of air and noise) when setting up routes and speed limits.

2.3 *Lleida municipality Pilots*

2.3.1 Road Incidents and Accessibility (Mobile App)

Abstract
The Road Incidents and Accessibility pilot consists of a mobile application with two main functionalities. The first one is checking, in real time, information about incidents in the city, like traffic accidents or closed streets because of public works. The second one is finding accessible services for people with mobility issues. Places are classified according to their category and to features making easier for the user the search for a convenient place.

Technologies used on pilot	**Data-sources silos**
• Open-DAI platform • REST/JSON • Python • Django framework • Javascript • Titanium framework	• Road incidents • City Facilities • Public transport • Accessibility

This pilot aims to publish the data silos detailed on the document D5.1 using the Open-DAI infrastructure and technologies. Once APIs are available the pilot aims to develop a specific utility to use this data and prove the use case detailed also on D5.1.

The Pilot development consists on add APIs on Open-DAI platform from silos detailed and a mobile application development addressed to the Barcelona citizens. The main goal is to achieve a useful service using the APIs opened with Open-DAI project.

The service developed to prove the data utility is a mobile app to give road-incidents and public facilities data on real-time to the citizens.

2.3.1.1 Pilot development

In order to develop the application “Road Incidents and Accessibility” in a service, we define a solution composed by a Mobile Application as a user interface and a service back-end to attend the petitions from the mobile phone.

The smart phone application interacts with the server application described and offers a graphical user interface for the consumers. This mobile application is developed using a

web based cross-platform technology in order to facilitate the availability on the different mobile platforms.

The following diagram shows the deployment of the artefacts of the pilot:

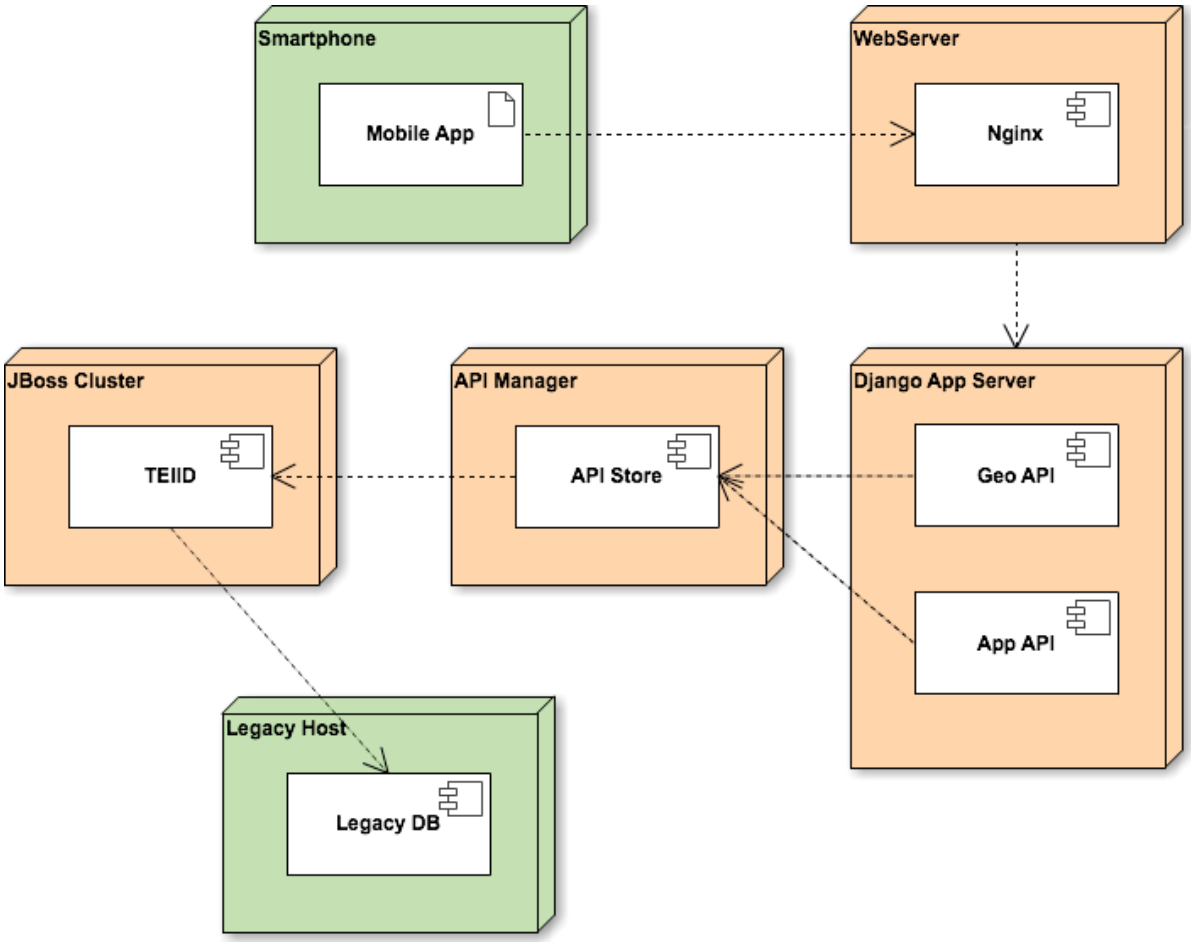


Figure 19: Block diagram of road incidents and accessibility

In the diagram there are two types of artefacts. The orange nodes are the Open-DAI platform nodes, while the green ones are the external nodes.

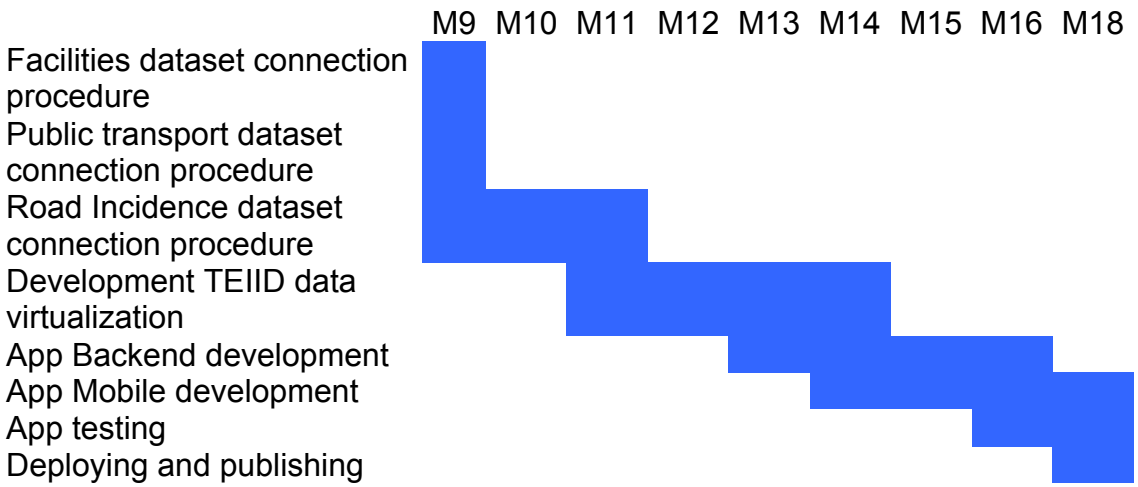
Describing the artefacts on detail:

- On Public administration host are localized the legacy data sources connected to the Open-DAI Platform.
- The first artefact as part of Open-DAI platform is the TEIID data virtualization system adapted to Barcelona city silos.
- Over the data virtualization the API manager provided by WSO2 framework offers access and control to the REST APIs publishing publicly a API store.
- Also part of Open-DAI platform is designed and developed using Django framework the back-end required by the mobile application. This back-end is composed by 2 artefacts:

- A Geo-server artefact developed using geodjango required by mobile app for the geospatial services such as geonaming reverse resolution and some different trigonometric functions.
- An application server developed using django with services and logic functions required by mobile application on purpose to light the computation on the mobile device.

- The access on the back-end services are provided and controlled by a proxy implemented with NGinx.
- At the end a cross platform mobile application is developed using Titanium framework as a client of the services and as an user interface application.

2.3.1.1.1 Followed development plan


M9 M10 M11 M12 M13 M14 M15 M16 M18
Facilities dataset connection procedure
Public transport dataset connection procedure
Road Incidence dataset connection procedure
Development TEIID data virtualization
App Backend development
App Mobile development
App testing
Deploying and publishing

2.3.1.1.2 Development results

The pilot results are mainly 2 different services, The APIs with public data available for citizens and 3rd party developers and a mobile application focused on adding a service on Barcelona city council public catalogue of services for the citizens.

APIS:

The APIs developed and published on this pilot is:

- Public Data Publication API: https://api.opendai.eu/store/apis/info?name=Data_Publication&version=1.0&provider=SpainAdmin

This APIs is explained and documented on the deliverable D6.1 as part of WP6.

App:

The result is an operational app application on iOS and Android platforms released on Lleida city council.

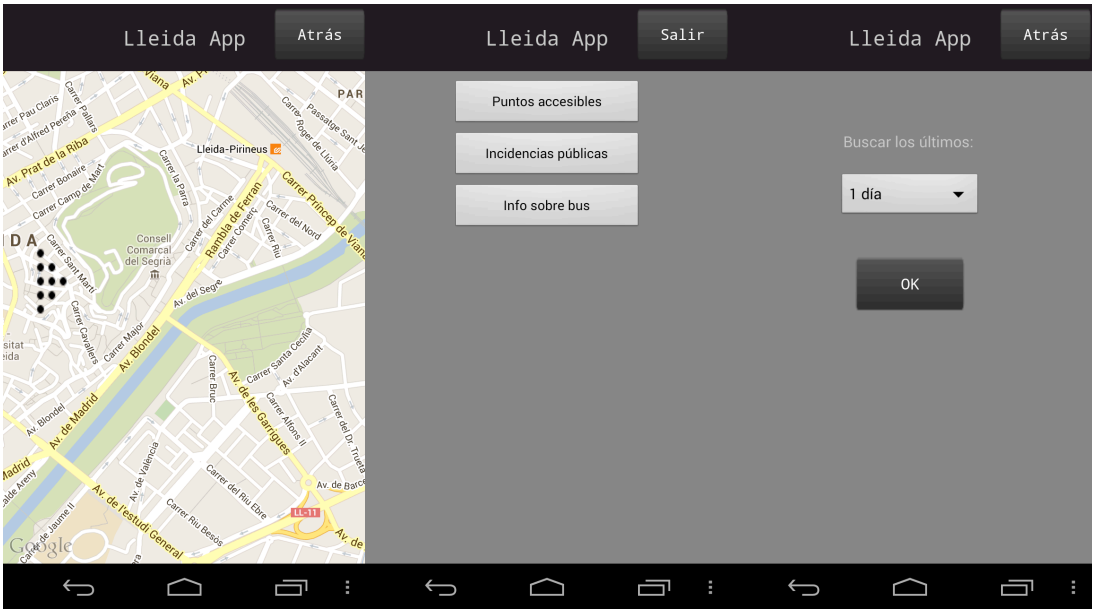


Figure 20: Screenshot Mobile App Road incidents and accessibility

2.3.1.2 Pilot documentation:

2.3.1.2.1 Pilot code repository

Github repository: https://github.com/open-dai/bcn-lleida-opendai-pilots

On Github repository we can found all the code necessary for Pilot divided on:_

- **jboss-teiid-apis** : the code needed to implement the TEIID data virtualization to connect to original data-sources (Silos)
- **mobile-apps** : the mobile application code.
- **web-geo-server** : the application back-end including the Geo-Server and the remote services needed by mobile app.

2.3.1.2.2 How-to install and deploy

The process to install this pilot is the same than the needed to deploy the Open-DAI platform, it is detailed on the WP4 deliverables. Inside this platform this Pilot uses:

- JBoss server and TEIID tool for the data virtualization.
- Virtual machines for the application servers
- API Manager for API publication.

As it has been explained this pilot is composed of a mobile app, a backend service over the Open-DAI infrastructure. To deploy these artefacts step-by-step we need to follow:

Data virtualization artefact:

The pilot calls to REST API from Open-DAI WSO2 infrastructure. This API is composed by a WAR (apilleida.war) for JBOSS and custom Translators for TEIID.

In order to deploy this API the source code and the JBOSS + TEIID environment are required. The data used by the pilot is virtualized on a virtual DB (lleida_base_vdb.vdb) deployed also in TEIID.

All configuration required is included on the war file and this should be deployed as a web application on JBOSS server included in this case on Open-DAI platform.

Application Back-end:

The Backend service is developed using Python-django framework. To deploy these services the source code and a server with Python 2.7 and Django 1.6.x are required.

The django applications run using WSGI technology behind a proxy web server. In our case, we used Gunicorn as WSGI provider and NGINX server as a proxy.

This application has to run on the same network where the TEIID artefact publishes the APIs. Then it is only needed to configure the IP address of this APIS on the django application settings.

Application Front-end:

The Mobile App is developed using Titanium framework. This is a cross-platform technology allowing the deployment on different App markets (iOS and Android).

The required items needed to deploy the apps are: the source code, the Titanium Studio Environment 3.1.x and the Apple and Android market environments.

Once the Android and iOS apps are built with the titanium studio is necessary follow the official methodology to deploy Apps on the Google and Apple markets.

2.3.1.2.3 User Manual

The app allows finding services nearby, in these case restaurants, with particular features, in this case an enabled exterior space. The user is able to select one or more features among a list that it is displayed with all the features registered. He also is able to select the type or types of services he is looking for, in this case a restaurant. Once the desired features are selected and the service types, a list of available services is presented to the user, either in list form or displayed on a map.

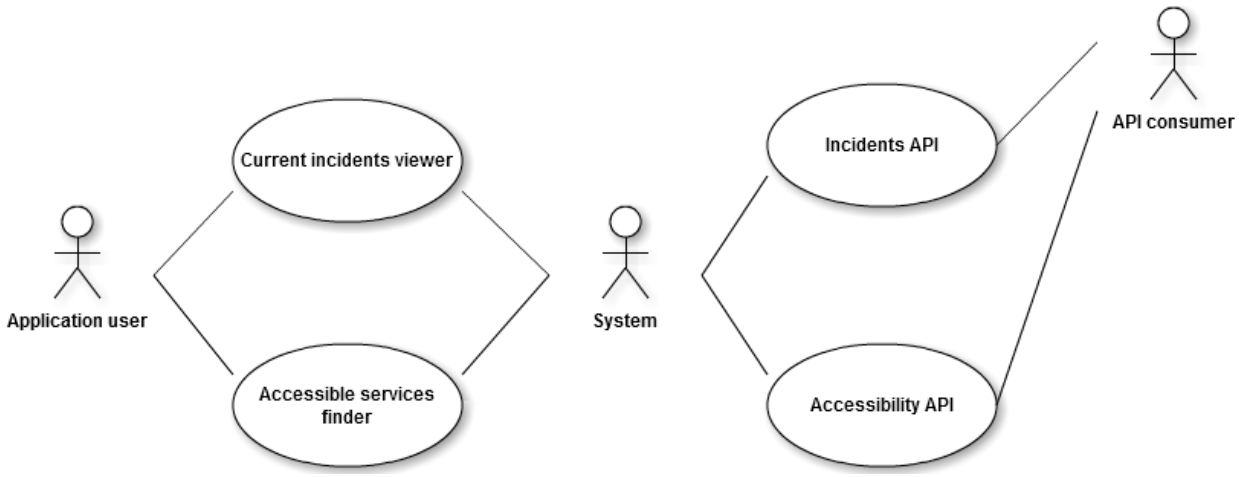


Figure 21: Use case Road incidents and accessibility

Using the Accessibility REST API, the application fetches, firstly, the lists of all the available service features and types and, secondly, the services that match those desired features.

2.3.2 Data publication (Web App)

Abstract	
The Data Publication pilot consists of a web application. The main functionality of this application shows the different services, municipal or not, and points of interest of the city, from public transports to tourism offices. This information is useful not only to the city tourists, but to the local population as well.	
Technologies used on pilot	**Data-sources silos**
• Open-DAI platform • REST/JSON • Python • Django framework • Javascript • Leaflet	Facilities by categories (Hospitality businesses, Educational centres, Bars and restaurants, Sport facilities, Local markets and services related, Tourism offices, Health care services, Municipal services, Non-municipal services, Mobility services information, Municipal bus stops)

This pilot aims to publish the data silos detailed on the document D5.1 using the Open-DAI infrastructure and technologies. Once APIs are available the pilot aims to develop a specific utility to use this data and prove the use case detailed also on D5.1.

The Pilot development consists on add APIs on Open-DAI platform from silos detailed and a mobile application development addressed to the Barcelona citizens. The main goal is to achieve a useful service using the APIs opened with Open-DAI project.

The service developed to prove the data utility is a Web application for the Lleida citizens to consume city data on real-time.

2.3.2.1 Pilot development

In order to develop the application correlation between environment and mobility in a service, we define a solution composed by a Web Application as a user interface and a service back-end to attend the petitions from the mobile phone.

The Web application interacts with the server application described previously and offers a graphical user interface for the consumers.

The following diagram shows the deployment of the artefacts of the pilot:

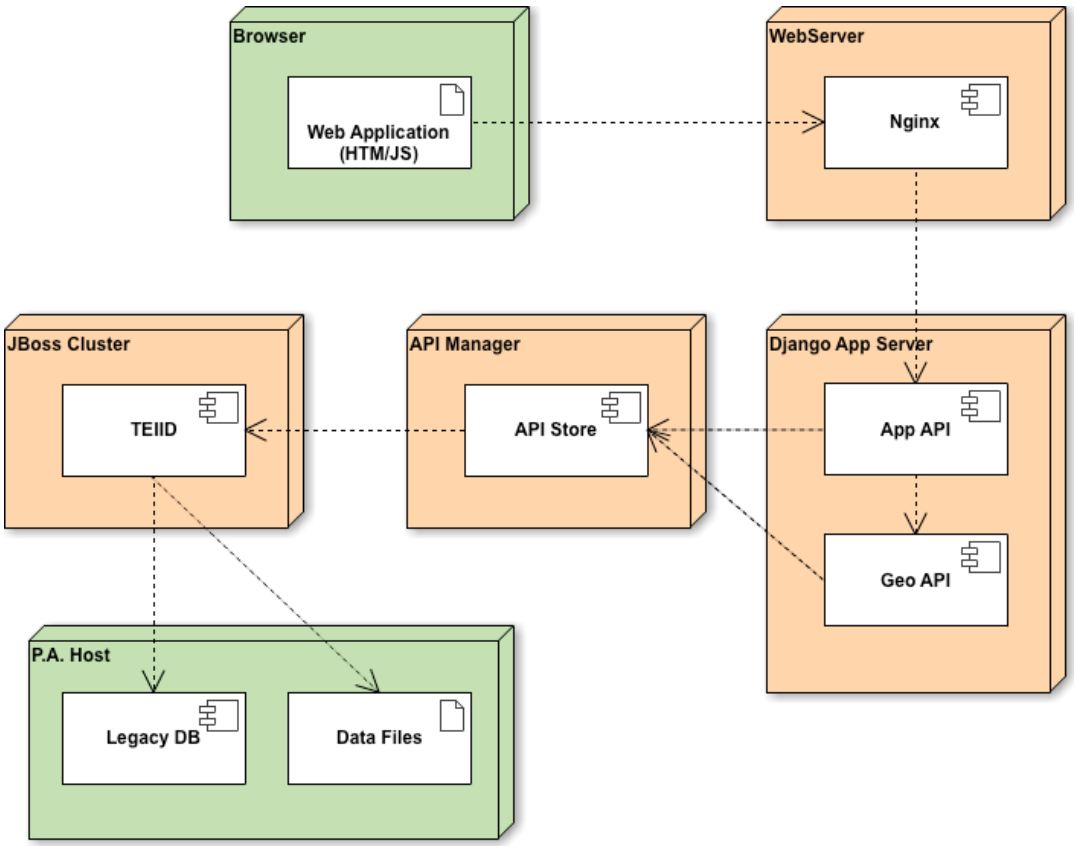


Figure 22: Block diagram Data publication

In the diagram there are two types of artefacts. The orange nodes are the Open-DAI platform nodes, while the green ones are the external nodes.

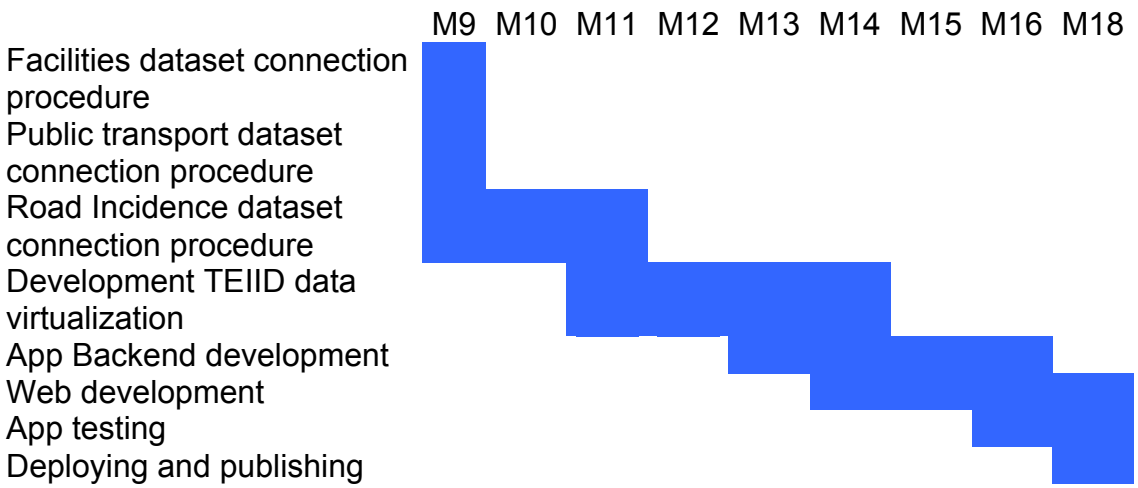
Describing the artefacts on detail:

- On Public administration host are localized the legacy data sources connected to the Open-DAI Platform.
- The first artefact as part of Open-DAI platform is the TEIID data virtualization system adapted to Barcelona city silos.
- Over the data virtualization the API manager provided by WSO2 framework offers access and control to the REST APIs publishing publicly an API store.
- Also part of Open-DAI platform is designed and developed using Django framework the back-end required by the mobile application. This back-end is composed by 2 artefacts:
    - A Geo-server artefact developed using geodjango required by the web application for the geospatial services such as geonaming reverse resolution and some different trigonometric functions.

- An application server developed using django with services and logic functions required by the web application on purpose to light the computation on the web browser.

- The web resources and the access on the back-end services are provided and controlled by a proxy implemented with NGinx.
- At the end the web application runs on a Web browser providing a user interface for the user.

2.3.2.1.1 Followed development plan


M9 M10 M11 M12 M13 M14 M15 M16 M18
Facilities dataset connection procedure
Public transport dataset connection procedure
Road Incidence dataset connection procedure
Development TEIID data virtualization
App Backend development
Web development
App testing
Deploying and publishing

2.3.2.1.2 Development results

The pilot results are mainly 2 different services, The APIs with public data available for citizens and 3rd party developers and a web application focused on adding a service on Lleida city council catalogue of services for internal public authorities.

APIS:

The APIs developed and published on this pilot is:

- Public Data Publication API: https://api.opendai.eu/store/apis/info?name=Data_Publication&version=1.0&provider=SpainAdmin

This APIs is explained and documented on the deliverable D6.1 as part of WP6.

App:

The result is an operational web app application published on this side:

http://lleida.opendai.eu

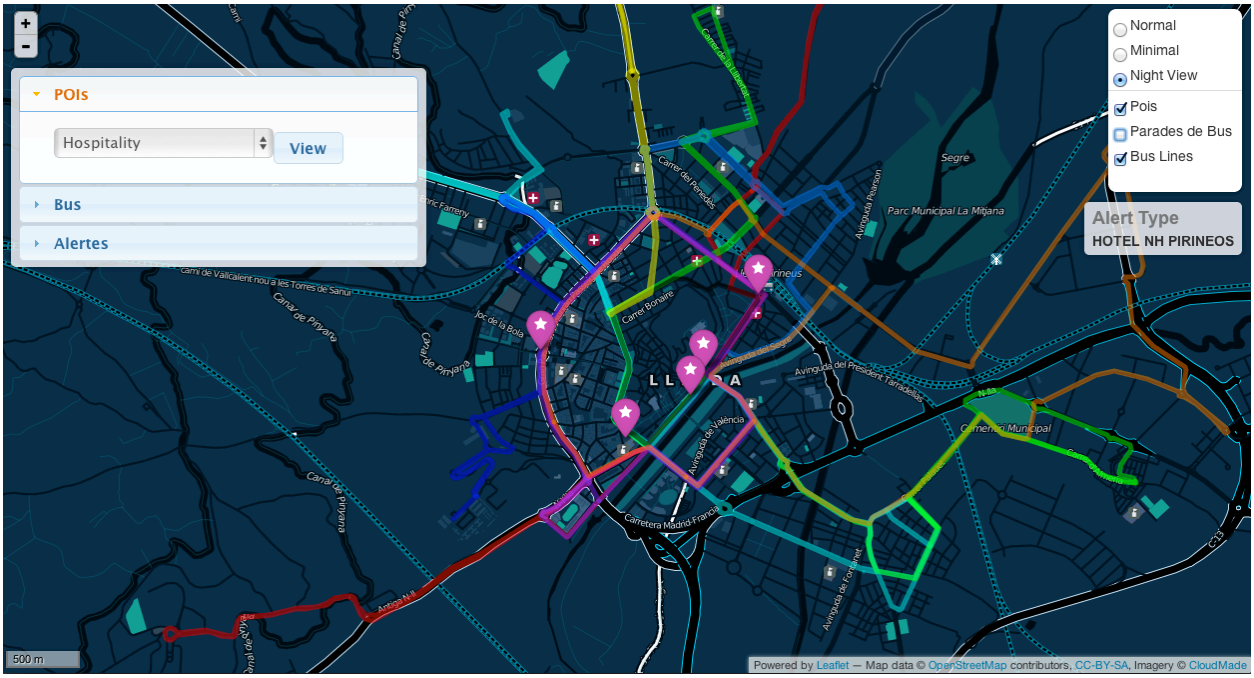


Figure 23: Screenshot Data Publication

2.3.2.2 Pilot documentation:

2.3.2.2.1 Pilot code repository

Github repository: https://github.com/open-dai/bcn-lleida-opendai-pilots

On Github repository we can found all the code necessary for Pilot divided on:

- **jboss-teiid-apis** : the code to implement the TEIID data virtualization to connect to original data-sources (Silos)
- **web-geo-server** : the application back-end including the Geo-Server and the web resources needed by the web app.

2.3.2.2.2 How-to install and deploy

The process to install this pilot is the same than the needed to deploy the Open-DAI platform as it is detailed on the WP4 deliverables. Inside this platform this Pilot uses:

- JBoss server and TEIID tool for the data virtualization.
- Virtual machines for the application servers
- API Manager for API publication.

As it has been previously explained this pilot is composed of a mobile app, a backend service over the Open-DAI infrastructure. To deploy these artefacts step-by-step we need to follow:

Data virtualization artefact:

The pilot calls to REST API from Open-DAI WSO2 infrastructure. This API is composed by a WAR (apilleida.war) for JBOSS and custom Translators for TEIID.

In order to deploy this API the source code and the JBOSS + TEIID environment are required. The data used by the pilot is virtualized on a virtual DB (lleida_base_vdb.vdb) deployed also in TEIID.

All configuration required is included on the war file and this should be deployed as a web application on JBOSS server included in this case on Open-DAI platform.

Application Back-end:

The Backend service is developed using Python-django framework. To deploy these services the source code and a server with Python 2.7 and Django 1.6.x are required.

The django applications run using WSGI technology behind a proxy web server. In our case, we used Gunicorn as WSGI provider and NGINX server as a proxy.

This application has to run on the same network where the TEIID artefact publishes the APIs. Then it is needed to configure the IP address of this APIS on the django application settings.

Application Front-end:

The Mobile App is developed using Titanium framework. This is a cross-platform technology allowing the deployment on different App markets (iOS and Android).

Required items to deploy the apps are: the source code, the Titanium Studio Environment 3.1.x and the Apple and Android market environments.

Once the Android and iOS apps are built with the titanium studio is necessary to follow the official methodology to deploy Apps on the Google and Apple markets.

2.3.2.2.3 User Manual

With the web application the user is able to search different kind of city data, for instance: bus routes and stops nearby his location or nearby a manually provided address. The bus routes and stops are displayed on a map of the city.

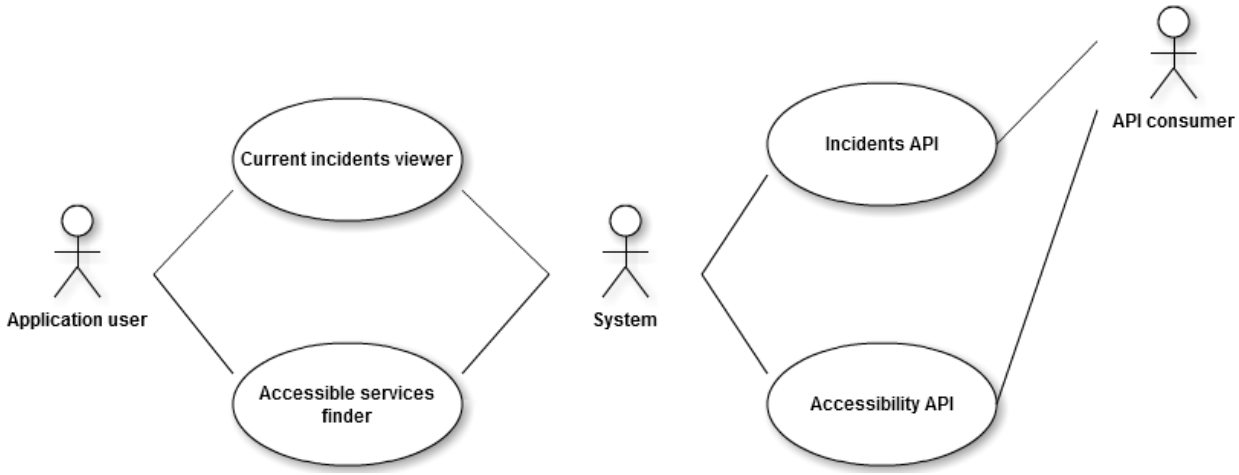


Figure 24: Use case Data publication

Using the City Bus Routes REST API, the application fetches the coordinates of the bus routes and stops nearby the ones.

2.4 Ordu municipality Pilots

2.4.1 City Dynamics (Mobile App)

Abstract	
City Dynamics provides a set of information that is designed to simplify the life in Ordu and it enables citizens and 3rd party developers to access daily information easily. Currently in Ordu, this type of information/content is only managed and generated by the Public Authority and citizens are only able to get information using the Call Center.	
Technologies used on pilot	**Data-sources silos**
• Open-DAI platform • WSO2 ESB Server • Geoserver • TEIID data repositories • Java SDK • Android SDK	• City facilities • Street directory • Points of interest • Reports

2.4.1.1 Pilot development

The following diagram shows the deployment of the artefacts.

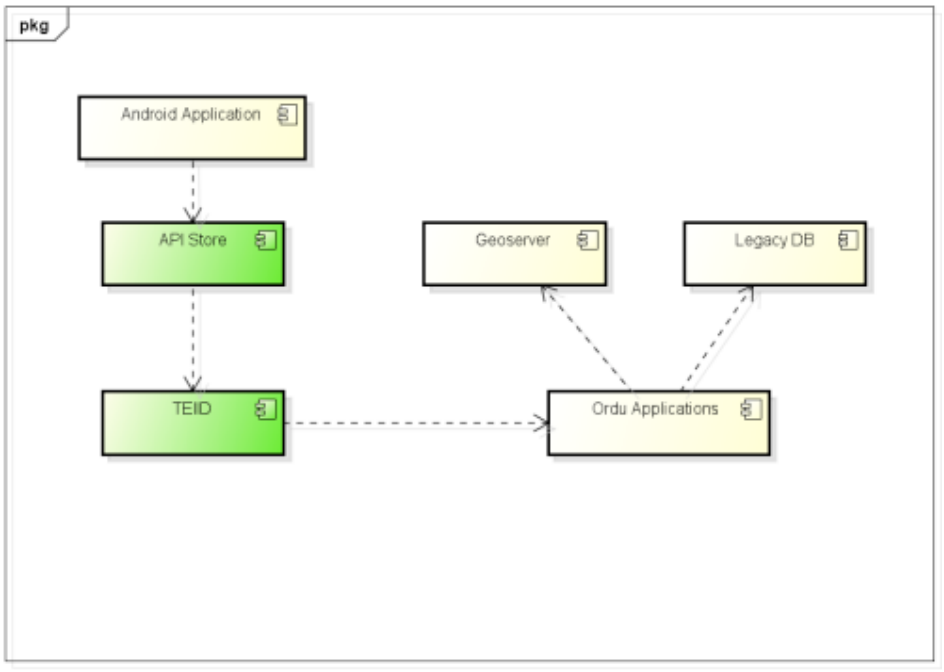


Figure 25: Block diagram City Dynamics

The smartphone takes the user input and choices. Then the data takes a way to legacy DB and Geoserver as stated above. Ordu Applications artefact consists of SOAP webservices and logic of the application.

Based on the use-cases defined in the technical description, citizens are able to start two different workflows. Call Center workflow and Citizen self-service workflow is given in Figure below. According to the Call Center workflow the citizen calls Ordu Call Center and asks for the closest garbage collection hours by location and by time. The Call Center Agent opens the map, searches the address of the citizen according to the given information. The Agent then finds the closest can to the citizen and shares the garbage collection hours information with the citizen. In the second workflow the citizen opens his or her map on the device, finds the closest can and get garbage collection hours information without the need of an agent.

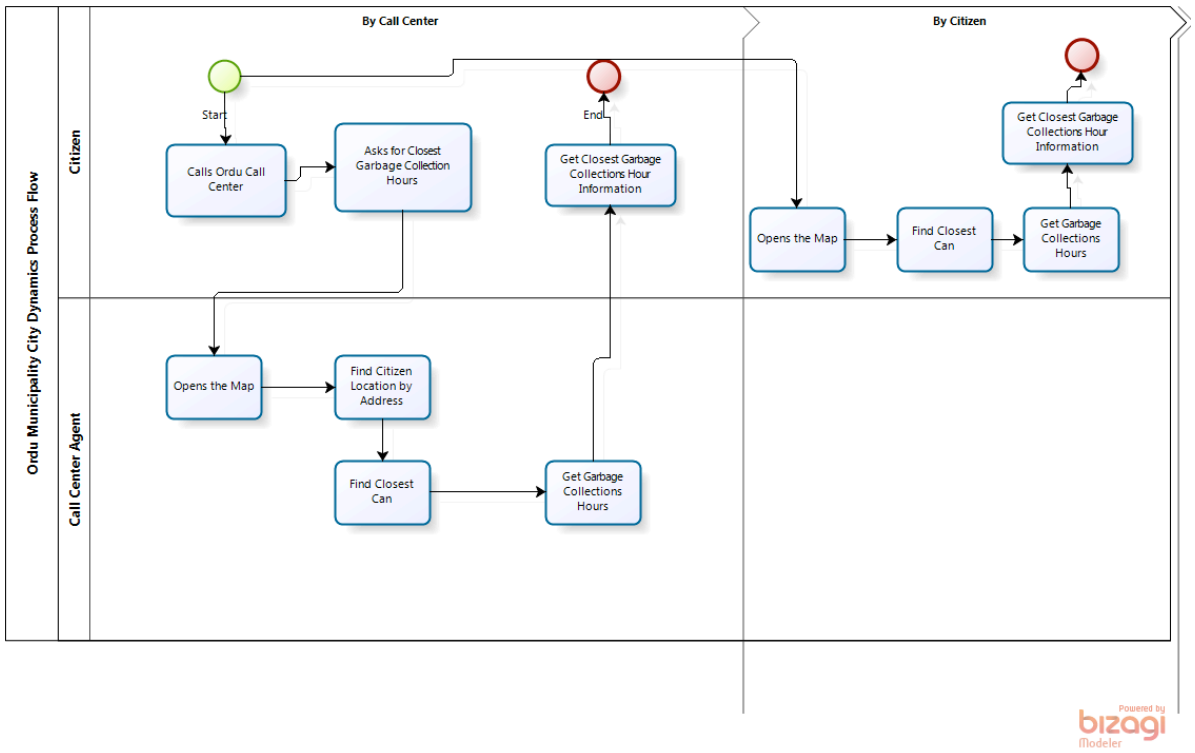


Figure 26: City Dynamics Workflow-I

Here, we present the screenshots of sample SOAP-UI requests and their expected results for our basic webservice operations which are presented in the workflow diagram above.

-listTheCansByStreet: In this sample request, we sent a “mahalle_id” (neighbourhood) value for querying the cans and their information which are available in that neighbourhood. Expected outcome of these operations is that we can get each of garbage cans’ detailed informations such as type, company, street, district info etc.

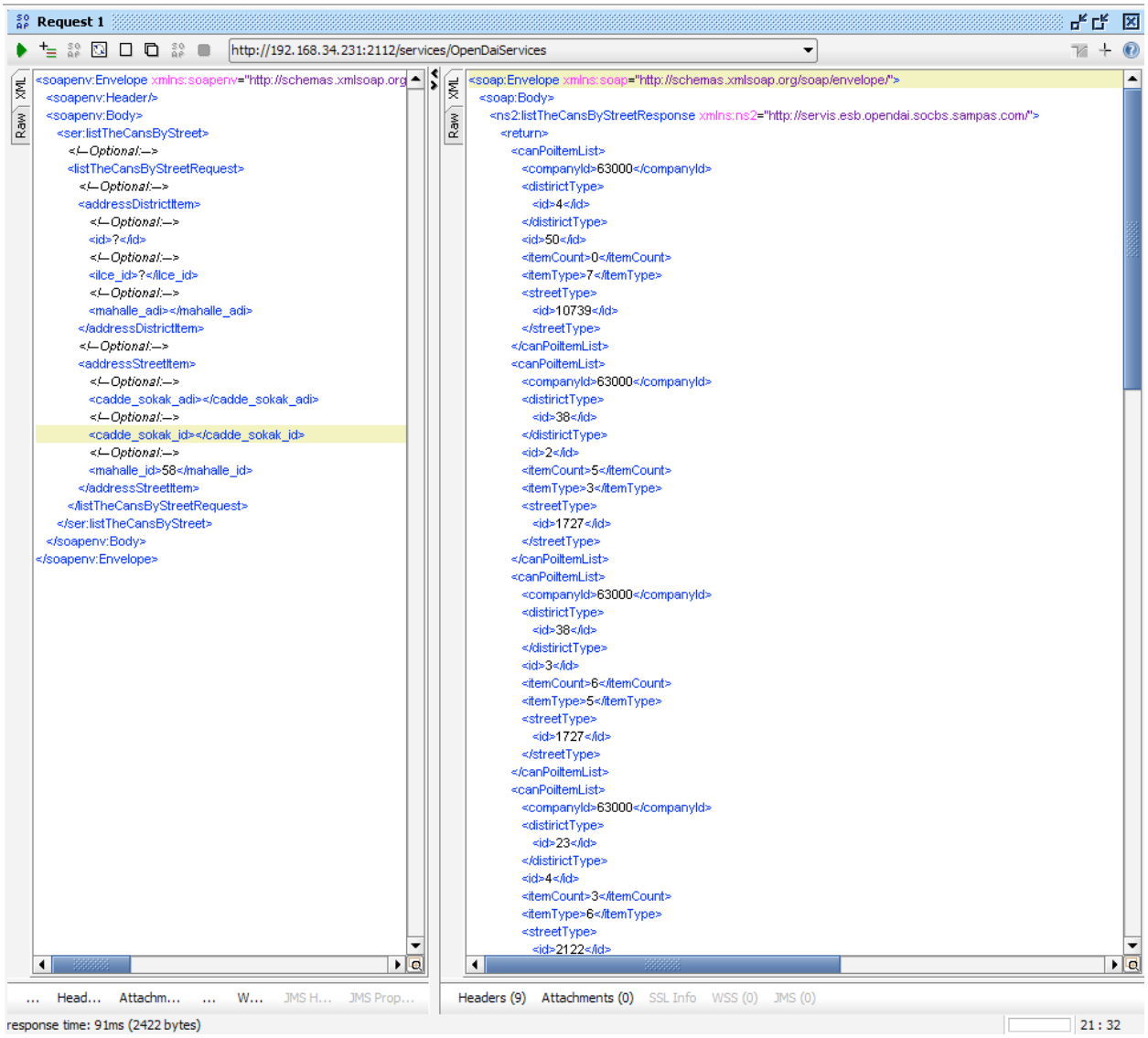


Figure 27: Screenshot Sample request

-getGcHoursForCan: In this operation, we sent a “mahalle_id” (neighbourhood) value again to get detailed day and hour values for the garbage cans in that area. As it can be seen, the result contains dates, starting and finishing times and other useful information about these cans.

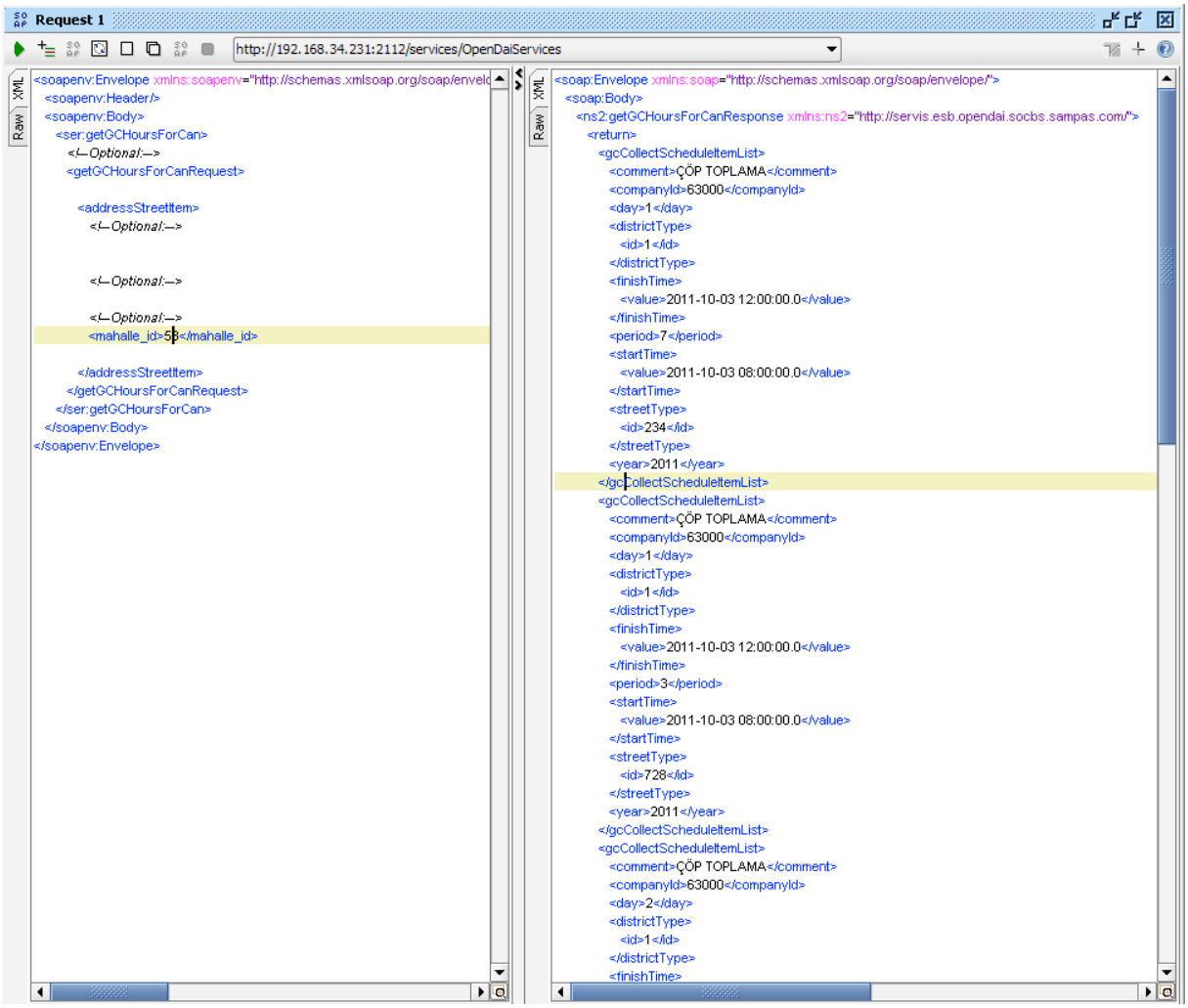


Figure 28: Screenshot result

The City Dynamics Open Data model also has two other different actors based on the goals defined previously. The System Admin in the Ordu Municipality is the one who is managing the system and the data. The System Admin is able to publish Web Map Service (WMS) and Web Feature Service (WFS) information using GeoServer in the Open Data Cloud platform. The System Admin is also able to update address and can Information including garbage collection hours to the system. Here update operation includes all types of creation, modification and deletion operations.

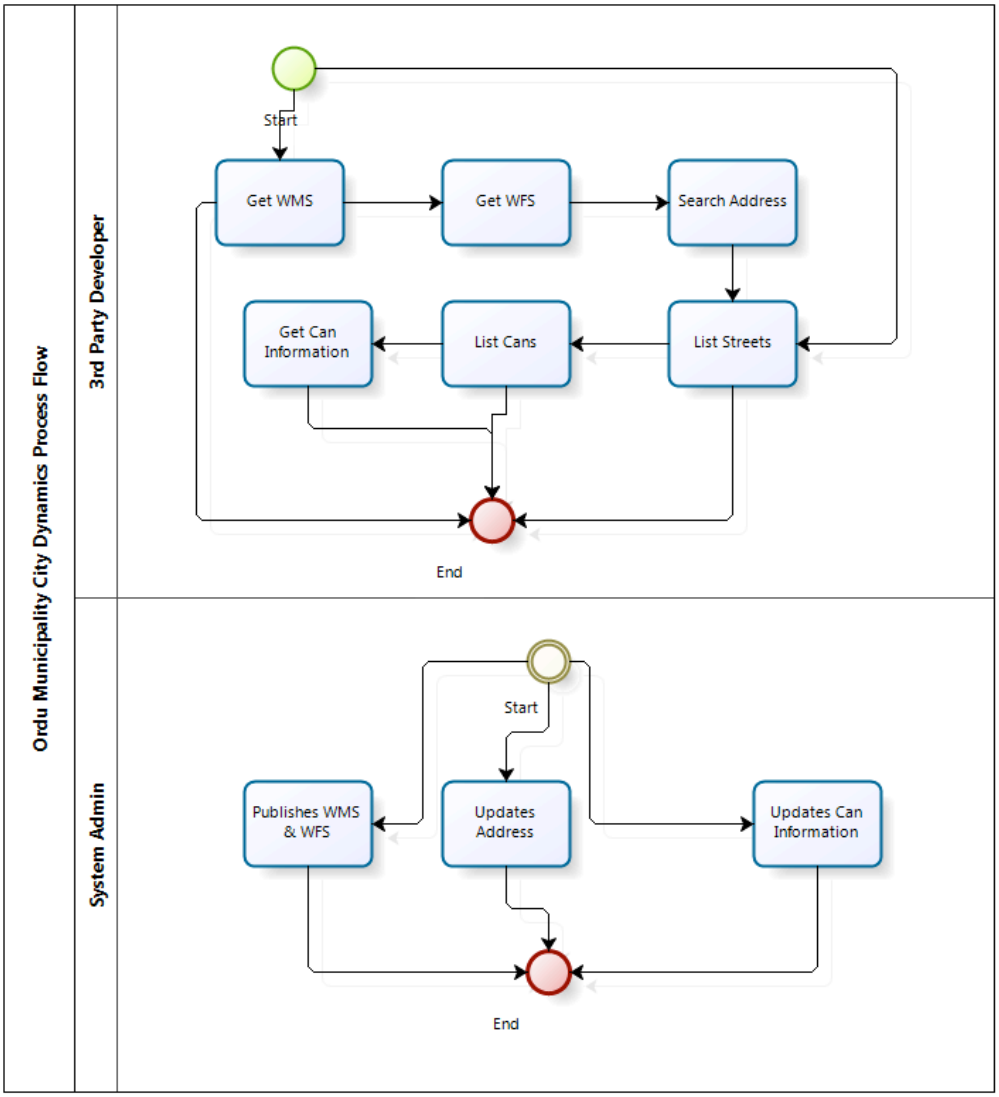


Figure 29: City Dynamics Workflow-II

Here, we present the screenshots of sample SOAP-UI requests and their expected results for our basic webservice operations which are presented in the workflow diagram above.

- listTheStreets: In this operation, all the available streets in that area is listed in the response. We can get name and id values. Also it can hold the other useful information about that street such as length and width.

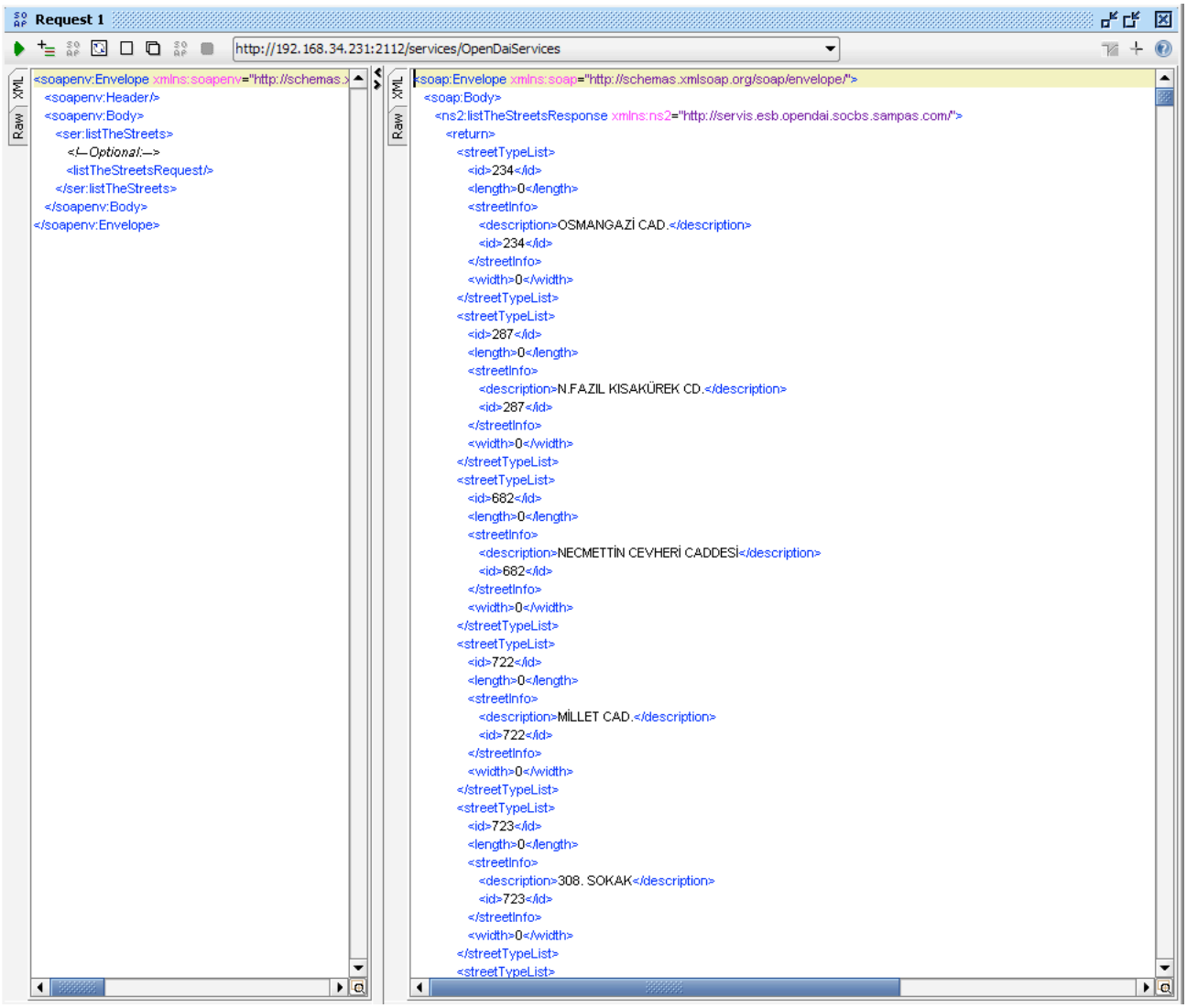


Figure 30: Screenshot list)

- getGCListByType: This operation sends a garbage can type as a parameter and gets the information about the cans which fit the given type. This return information has the coordinates, start and finish hours for each of these cans. In the sample request below, we requested the cans in type of “Hastane” (hospital).

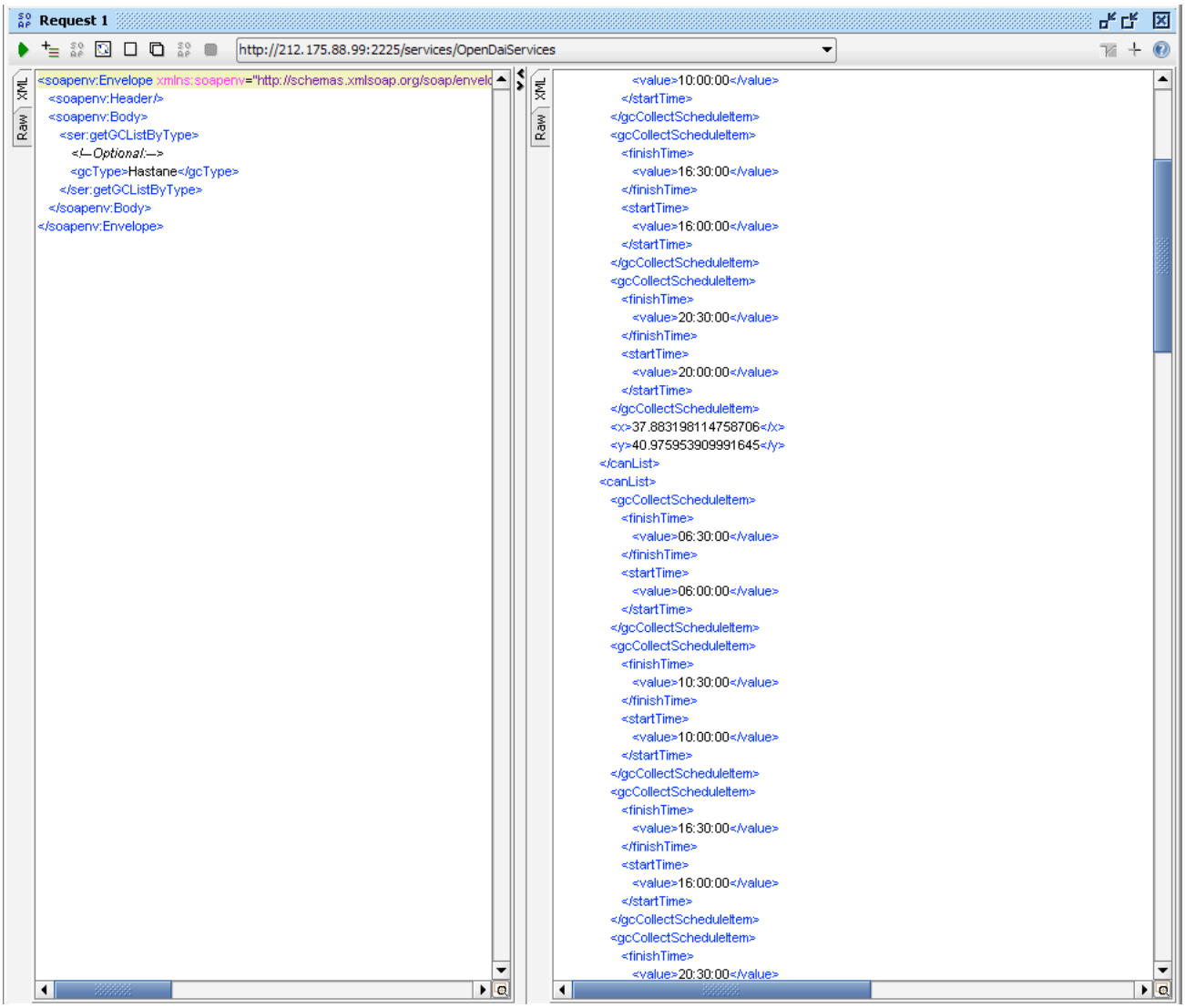


Figure 31: Screenshot information

-getInformationForCan: In this operation, the outcome consists of information about for a specific garbage can. In the sample request presented below, we asked information about the can with Id of 2 and the return has useful information like company, district & street info, itemType and itemCount.

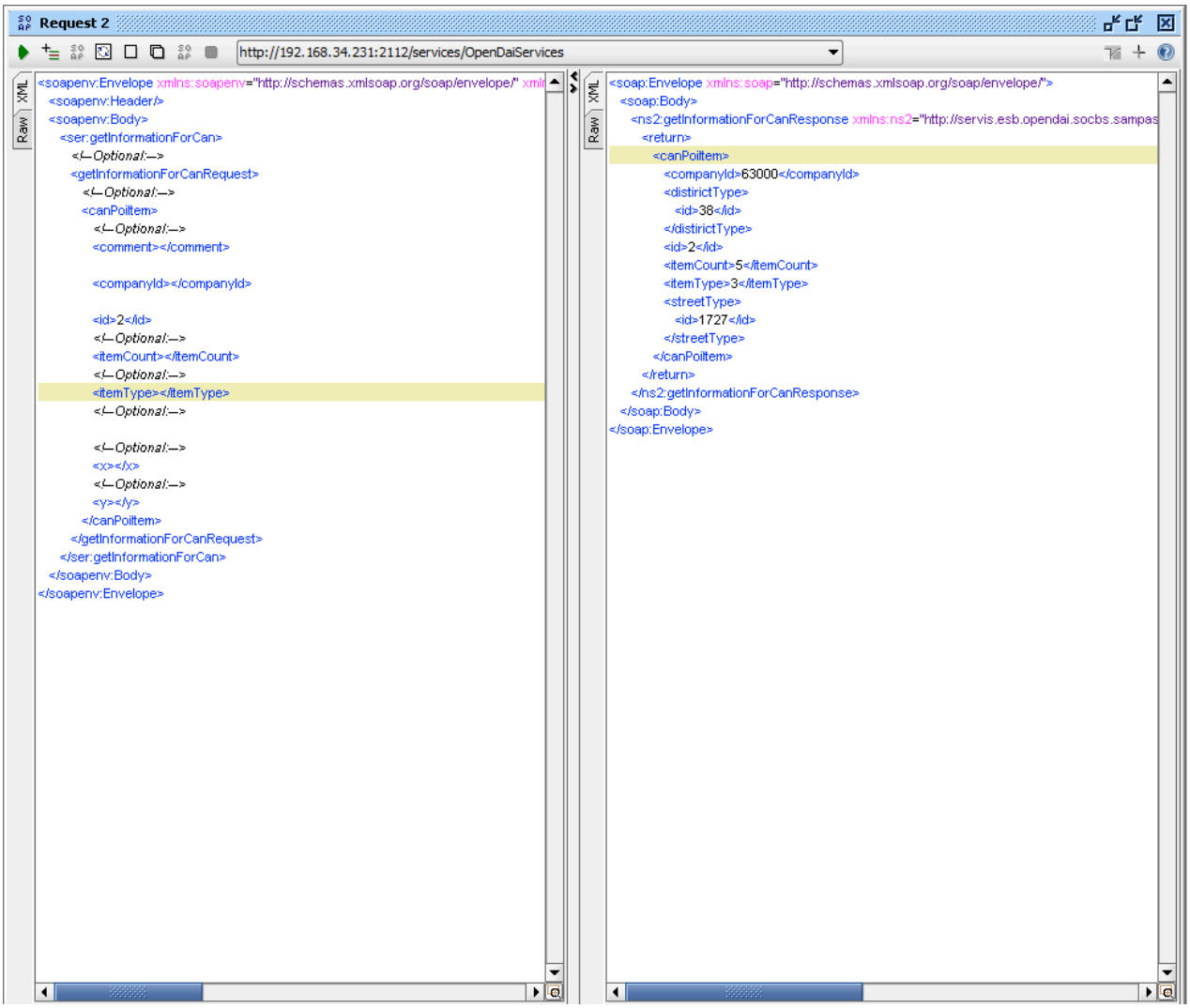


Figure 32: Screenshot getInformation

-queryCanType: By using this operation, one can see the available garbage can types and their respected Id's.

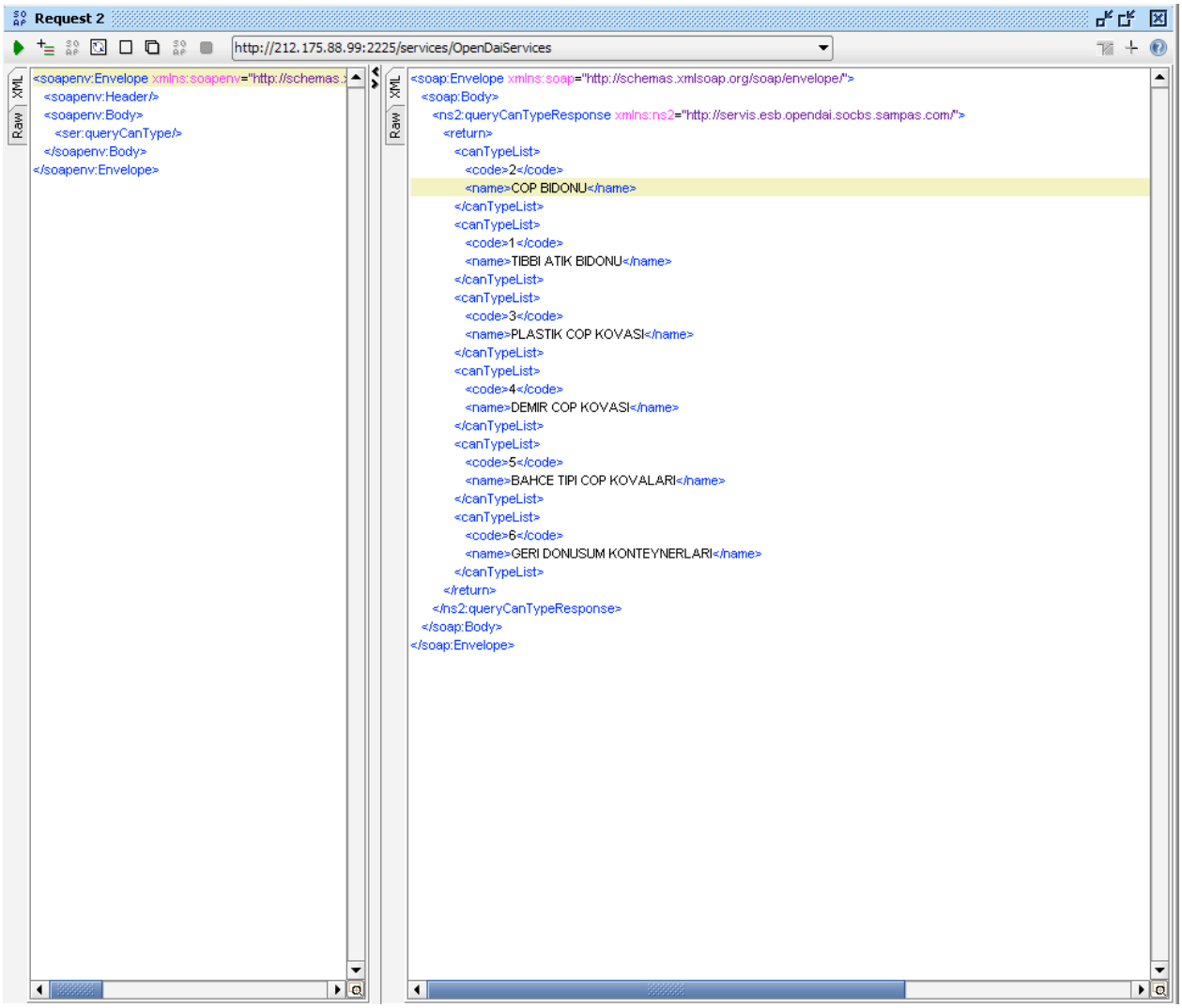
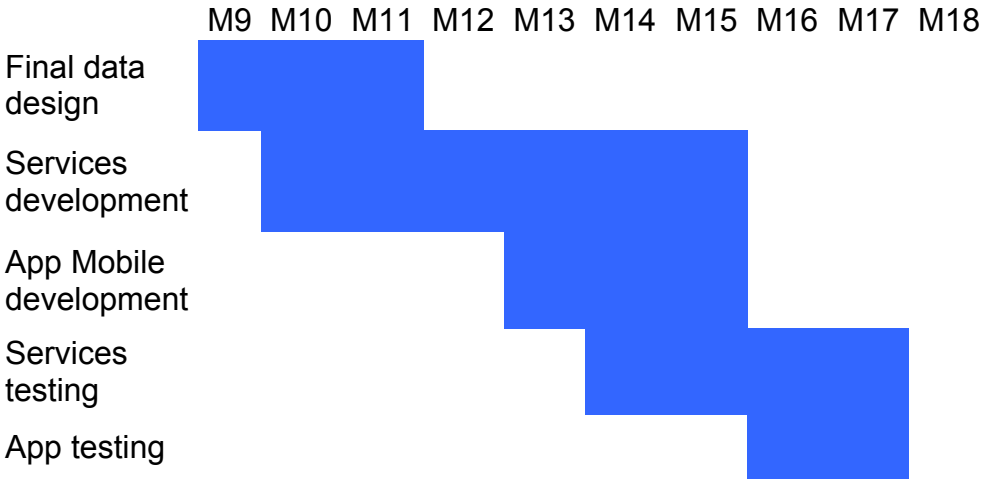


Figure 33: Screenshot query

2.4.1.1.1 Followed development plan



Deploying and publishing

2.4.1.1.2 Development results

OrduServices provide a set of information that is designed to simplify the life in Ordu and it enables citizens and 3rd party developers to access daily information easily. Currently in Ordu, this type of information/content is only managed and generated by the Public Authority and citizens are only able to get information using the Call Center. Secondly, these services provide different kinds of information about the nearest garbage bin according to its properties.

APIs:

https://api.opendai.eu/store/apis/info?name=OrduServices&version=1.0.0&provider=TurkeyAdmin

More information can be found on the deliverable D6.1 as part of WP6 and also on the documentations in the API Store documentation part.
Mobile Application:

An Android application was prepared to distribute the data and make shown in the map.

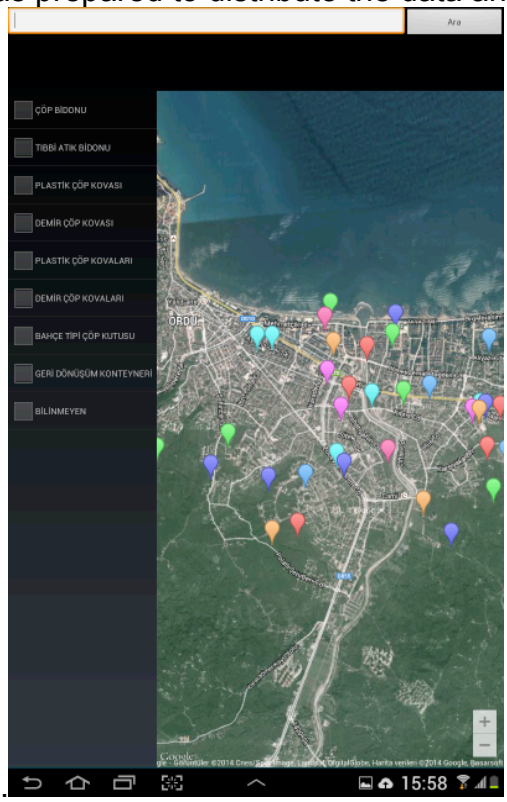


Figure 34: Mobile App image

In the application, user can select the types of the garbage bins to locate them on the map.

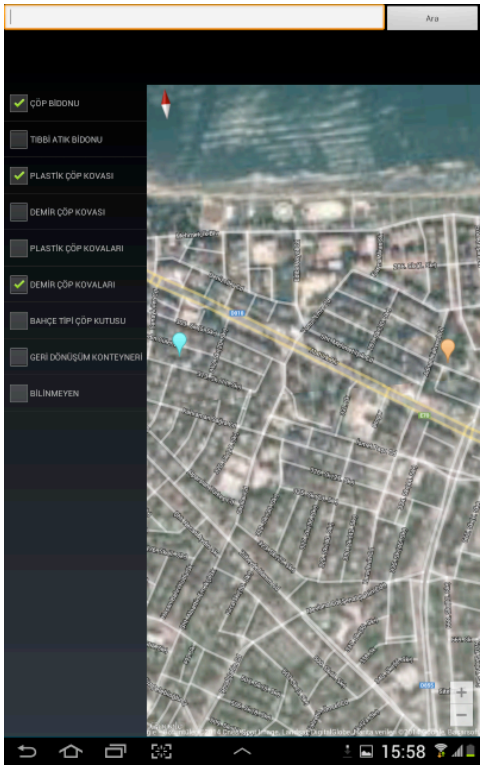


Figure 35: Mobile App image

2.4.1.2 Pilot documentation:

2.4.1.2.1 Pilot code repository

Github repository: https://github.com/open-dai/ordu-pilot-citydynamics

The source code of the CityDynamics project, which was coded in Java, is available in the repository. The respective packages and class files can be seen there.
There is a simple document named README.md which basically sums up the methods, their aim and usage. Also the main actions and their related webservice operations are available there.

2.4.1.2.2 How-to install and deploy

The Deployment Diagram for City Dynamics services in Open Data Platform is given as follows:

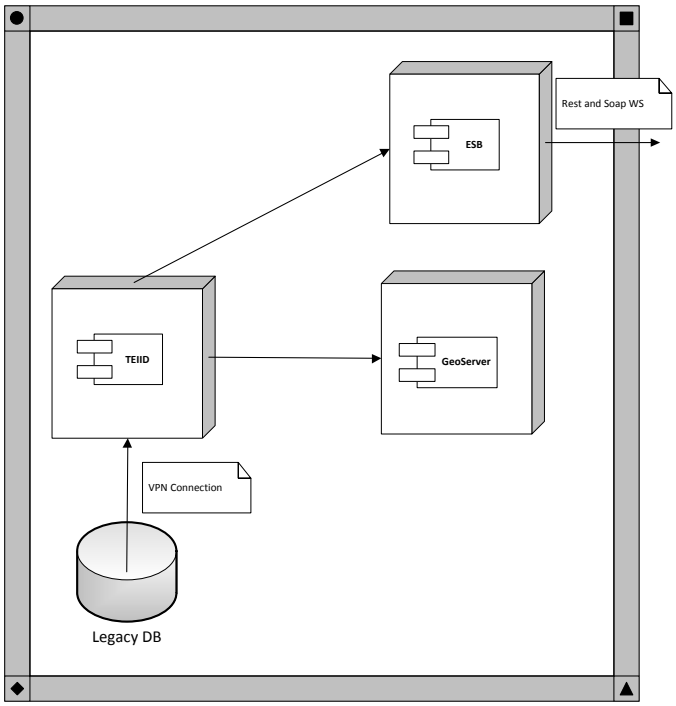


Figure 36: City Dynamics Deployment Diagram

2.4.1.2.3 User Manual

Use-Case 1: Get Garbage Collection Hours By Location – Users are able learn garbage collections hours for the closest cans using their application. This use-case defined the actions that can be taken by the citizen or the call center agent by location information. Geoserver is used for serving geographical data.

Use-Case 2: Get Garbage Collection Hours - Users are able learn garbage collections hours for the closest cans using their application. This use-case defined the actions that can be taken by the citizen or the call center agent without using location information.

Use-Case 3: 3rd Party Development – This use case identifies potential areas of 3rd party development for optimizing the garbage collection issues in the municipality. This provides actions that 3rd party developers are able to do using the opened data from the platform.

Use-Case 4: System Administration – System is controlled and managed by the system administrators in the PA. This use case provides actions that a system admin can take during the system runtime.

Use-Case 1: Get Garbage Collection Hours By Location

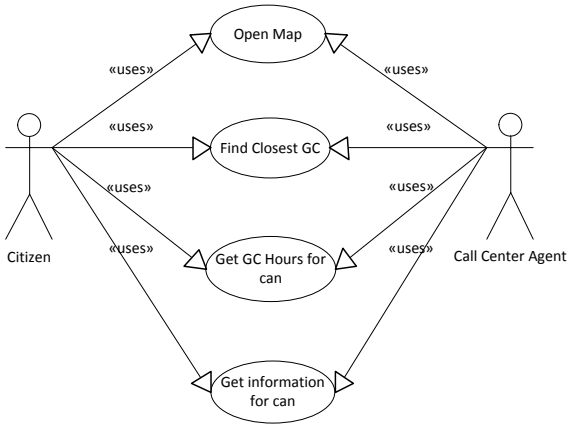


Figure 37: City Dynamics Use-Case1: Get Garbage Collection Hours By Location

Actions	Description
Open Map	User is able to open the map and navigate the map
Find Closest GC	User can find the closest can for gps position
Get GC Hours for can	User can find the schedule for the GC
Get information for can	User can find information for GC

Use-Case 2: Get Garbage Collection Hours

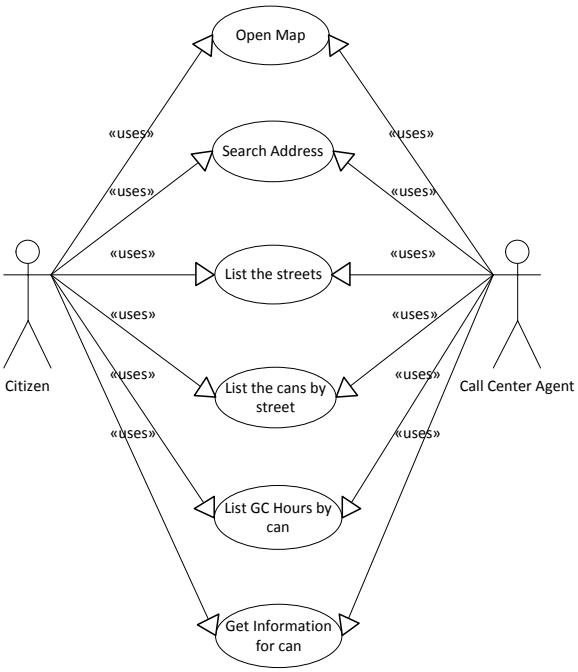


Figure 38: City Dynamics Use-Case2: Get Garbage Collection Hours

Actions	Description
Open Map	User is able to open the map and navigate the map
Search address	User is able to search for an address
List the streets	User can see the list of streets
List the cans by street	User can see the GC list for a street
List GC Hours by can	User can see the collection hours for a GC
Get information for can	User can see the information for a GC

Use-Case 3: 3rd Party Development

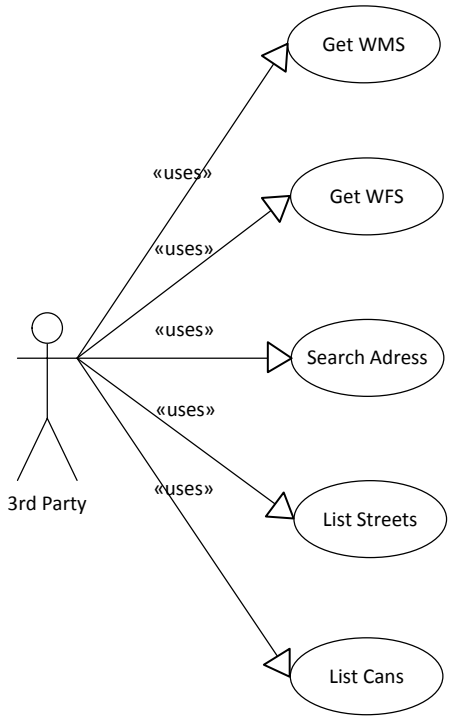


Figure 39: City Dynamics Use-Case3: 3rd Party Development

Actions	Description
Get WMS	User can reach the maps as images for an area
Get WFS	User can reach the vector spatial data for an area
Search address	User can search for an address
List Streets	User can see the list of streets
List cans	User can see the GB list for related street

Use-Case 4: System Administration

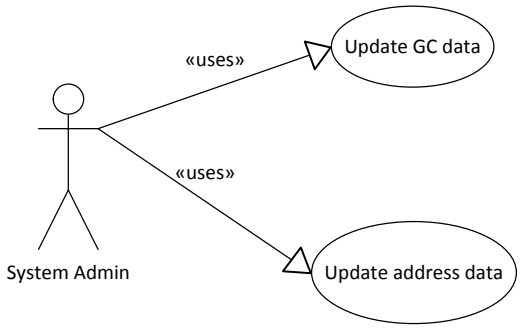


Figure 40: City Dynamics Use-Case4: System Administration

Actions	Description
Update GC data	Admin is able to create, update and delete GC data
Update address data	Admin can create, update and delete address data

2.4.2 Poi`s (Mobile App)

Abstract	
POI (Points of Interest) provides location information about interesting points around the user location. The citizen or the tourist, after locating his position on a map, may ask for a specific location such as the nearest taxi, pharmacies, mosques, museums, cultural places, the nearest train station or the user can also ask for the nearest hospital.	
Technologies used on pilot	**Data-sources silos**
• Open-DAI platform • WSO2 ESB Server • Geoserver • TEIID data repositories • Java SDK • Android SDK	• City facilities • Street directory • Points of interest • Reports

2.4.2.1 Pilot development

The following diagram shows the deployment of the artefacts.

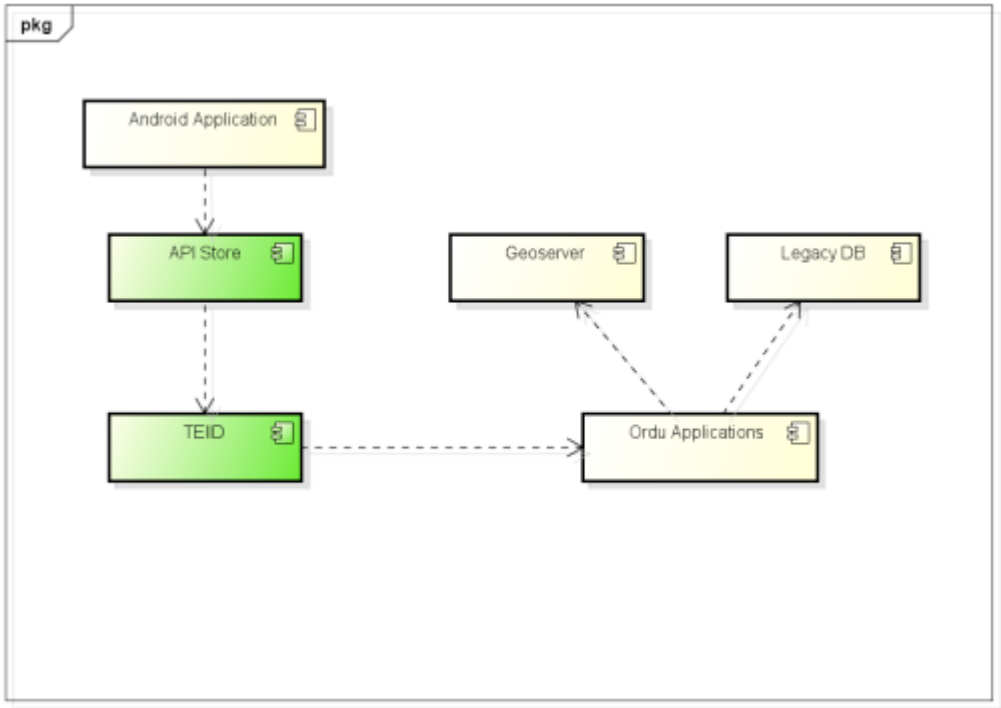


Figure 41: Block Diagram POIs

The smartphone takes the user input and choices. Then the data takes a way to legacy DB and Geoserver as stated above. Ordu Applications artefact consists of SOAP webservices and logic of the application.

The figure below shows the process flows according to the defined use-cases. Citizen or a Tourist is able to list POI types, show the POIs on the map, get POI list based on their types, list POIs according the their locations and given area and also get closest POI information based on their location. They will also be able to get routing information to the selected POIs. 3rd Party developers are able to develop POI based applications by the features provided from the Open Dai platform. The business process flow also shows the operations that a system admin can take during the management of the POI services.

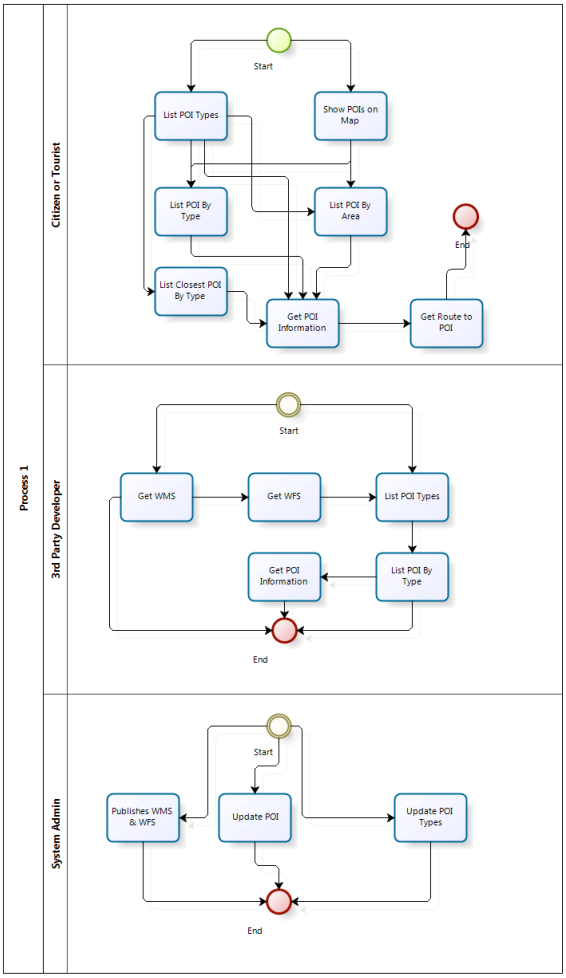


Figure 42 POI Workflow

Here, we present the screenshots of sample SOAP-UI requests and their expected results for our basic webservice operations which are presented in the workflow diagram above.

-listPOITypes: This operation returns the available POI types. As can be seen from the sample request presented below, POI types such as “hastane” (hospital), “eczane” (pharmacy) are listed.

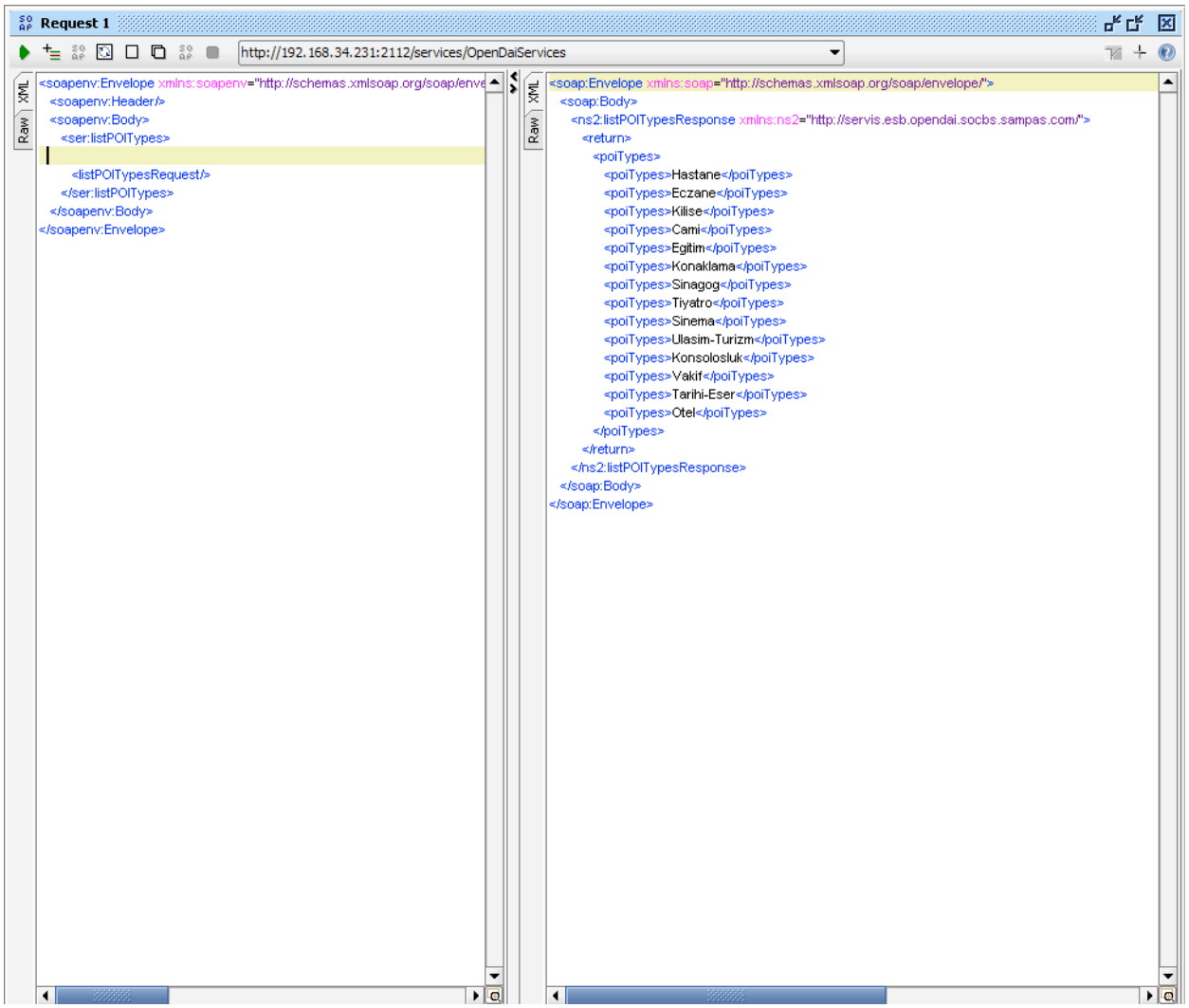


Figure 43: Screenshot sample request

-listPOIsByType: This operation queries and presents info about the POI locations for a given POI type. In our sample, we questioned the POI's in the type of "Eczane" (pharmacy) and the response gives information about the pharmacies with their coordinates.

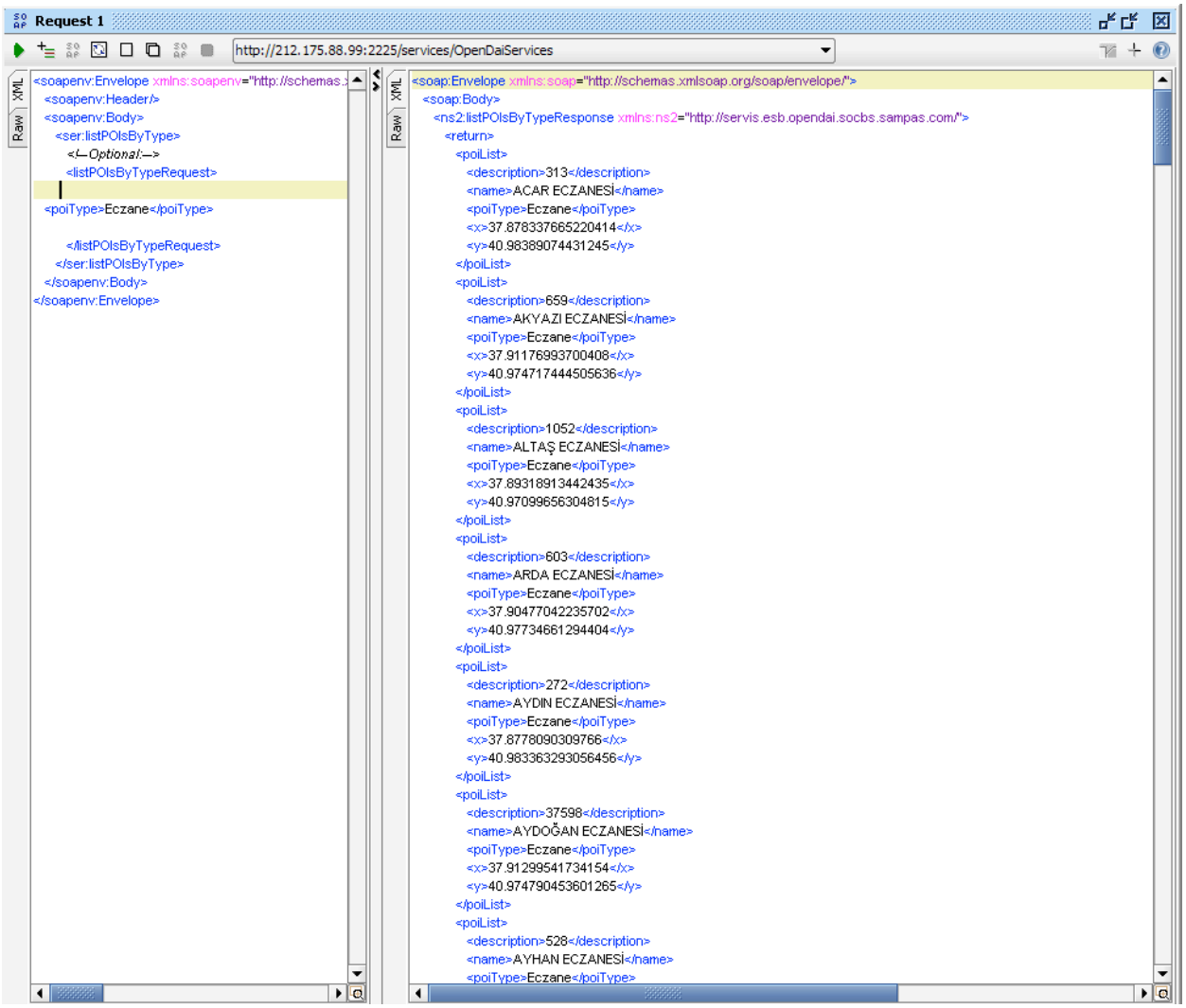


Figure 44: Screenshot request POIs

-queryClosestPOI: This operation finds and presents the closest POI for a given user coordinate and a POI type. In the sample presented below, we sent the coordinates and the POI type “Eczane” (pharmacy) we wish to find. As a a result, we got the closest pharmacy’s coordinates and name.

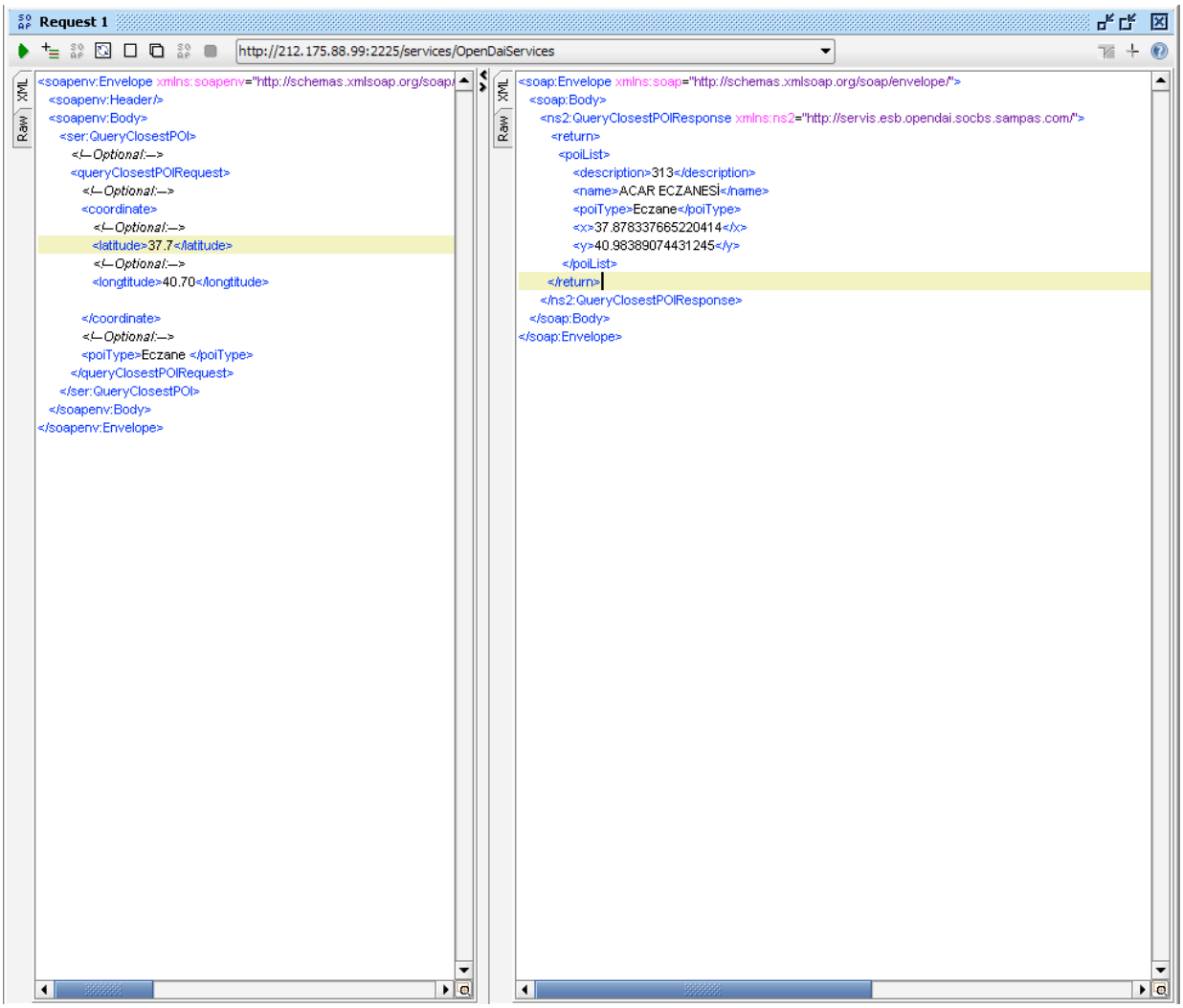


Figure 45: Screenshot request ClosestPOI

Also there are other helpful requests. Some of them are:

-listTheAddressCity: This operation gives all the cities and their Id's.

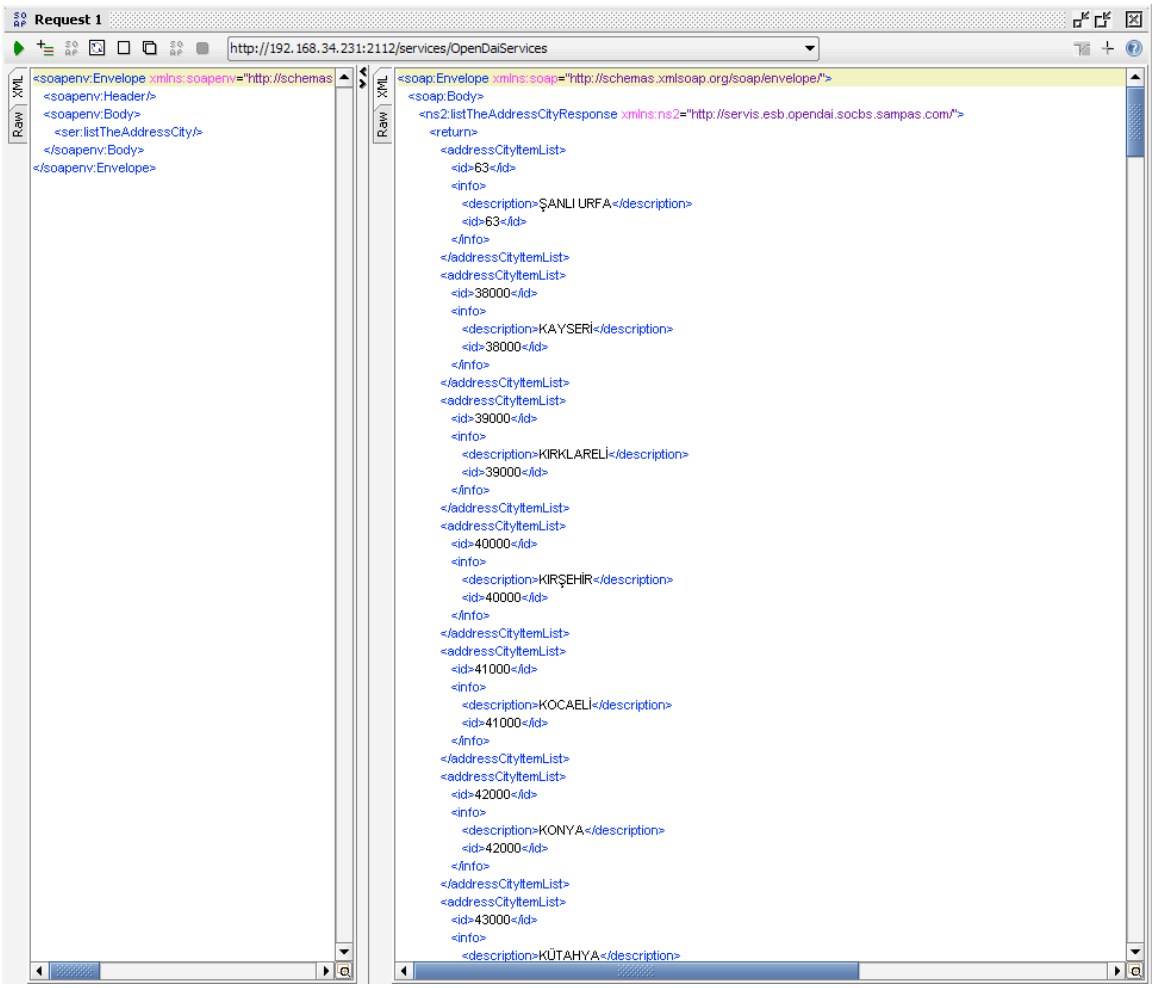


Figure 46: Screenshot list cities

- listTheAddressStreet: In this operation for a given city, district or neighbourhood, one can get detailed street information about this area. In the below example, we requested the info for the city which has the Id 63

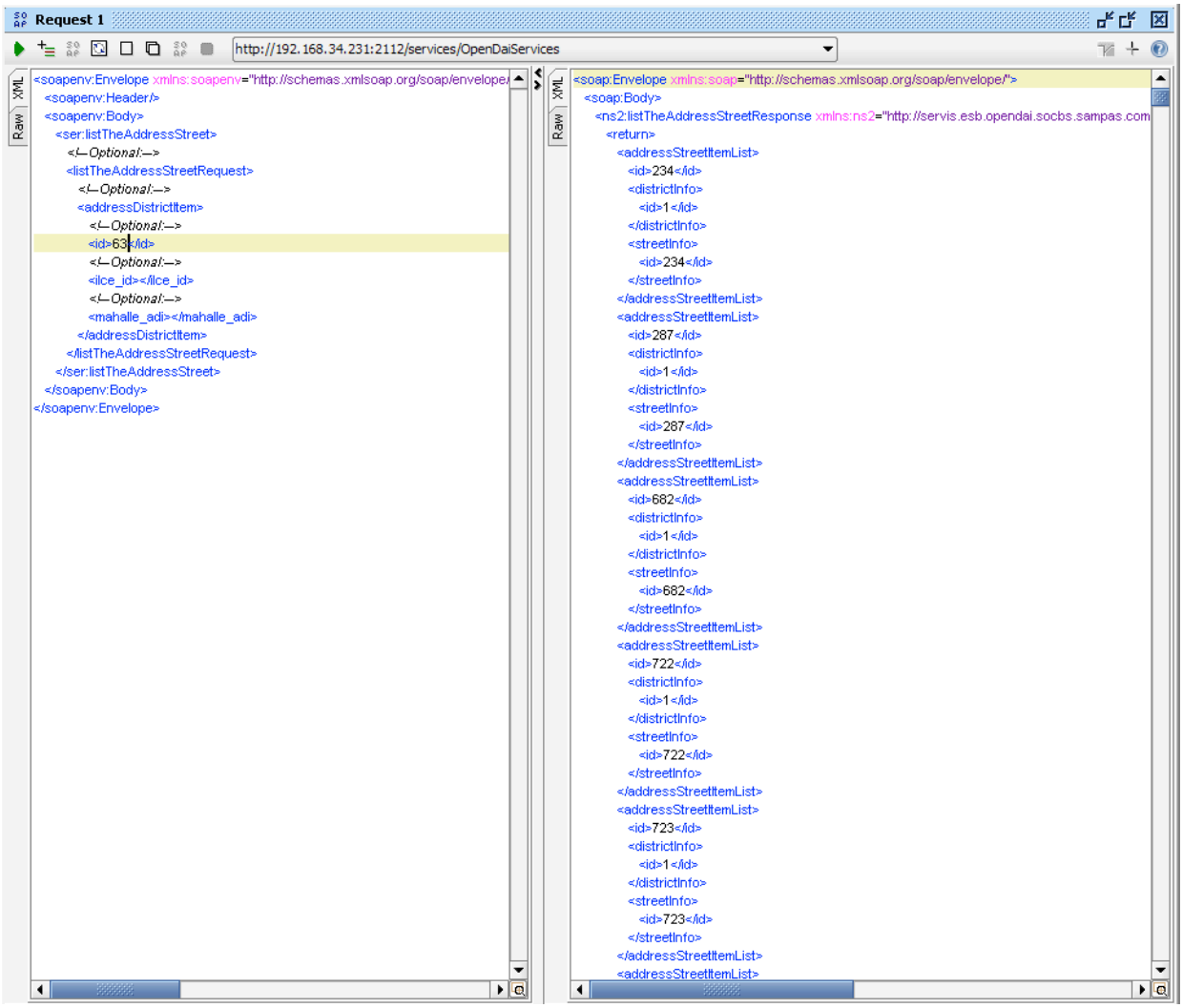
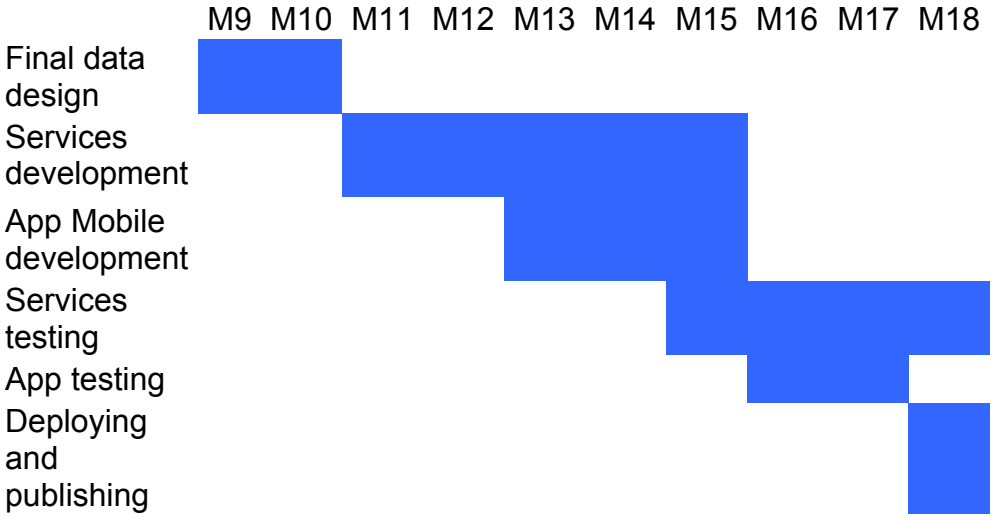


Figure 47: Screenshot request detailed information

2.4.2.1.1 Followed development plan



2.4.2.1.2 Development results

OrduServices provide a set of information that is designed to simplify the life in Ordu and it enables citizens and 3rd party developers to access daily information easily. Currently in Ordu, this type of information/content is only managed and generated by the Public Authority and citizens are only able to get information using the Call Center. Secondly, these services provide location information about interesting points around the user location. The citizen or the tourist, after locating his position on a map, may ask for a specific location such as the nearest taxi, pharmacies, mosques, museums, cultural places, the nearest train station or the user can also ask for the nearest hospital.

APIs:

https://api.opendai.eu/store/apis/info?name=OrduServices&version=1.0.0&provider=TurkeyAdmin

More information can be found on the deliverable D6.1 as part of WP6 and also on the documentations in the API Store documentation part.

App:
An Android application was prepared to distribute the data and make shown in the map. So that user can locate different kinds of places on the map according to its POI type.

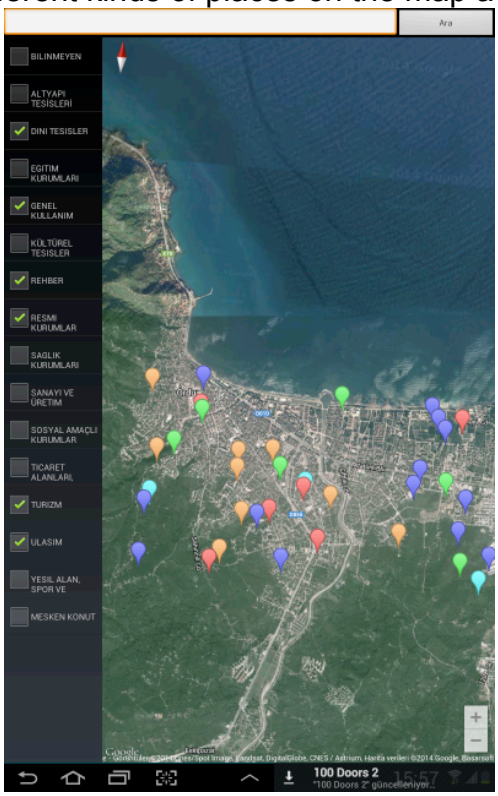


Figure 48: Screenshot Mobile App POIs

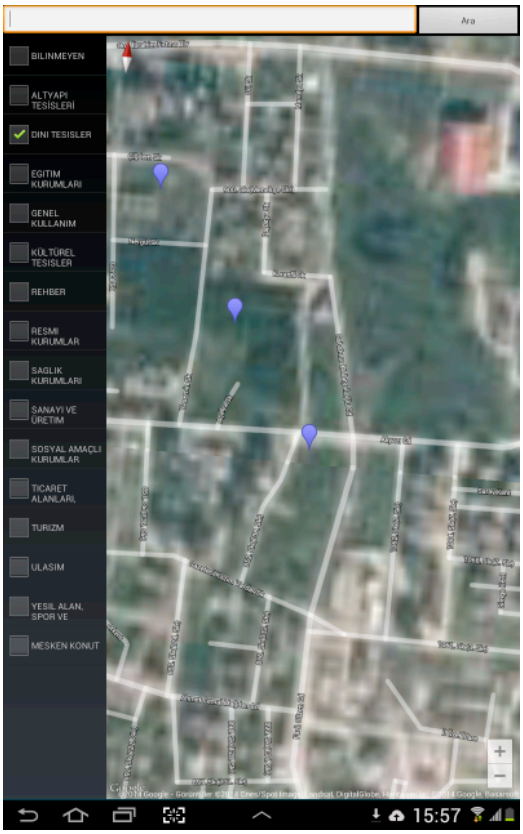


Figure 49: Screenshot Mobile App POIs (2)

2.4.2.2 Pilot documentation:

2.4.2.2.1 Pilot code repository

The source code of the POI project, which was coded in Java, is available in the repository. The respective packages and class files can be seen there.
In the repository, there is a simple document named README.md which basically sums up the methods, their aim and usage. Also the main actions and their related webservice operations are available there.

2.4.2.2.2 How-to install and deploy

The Deployment Diagram for POI services in Open Data Platform is given as follows:

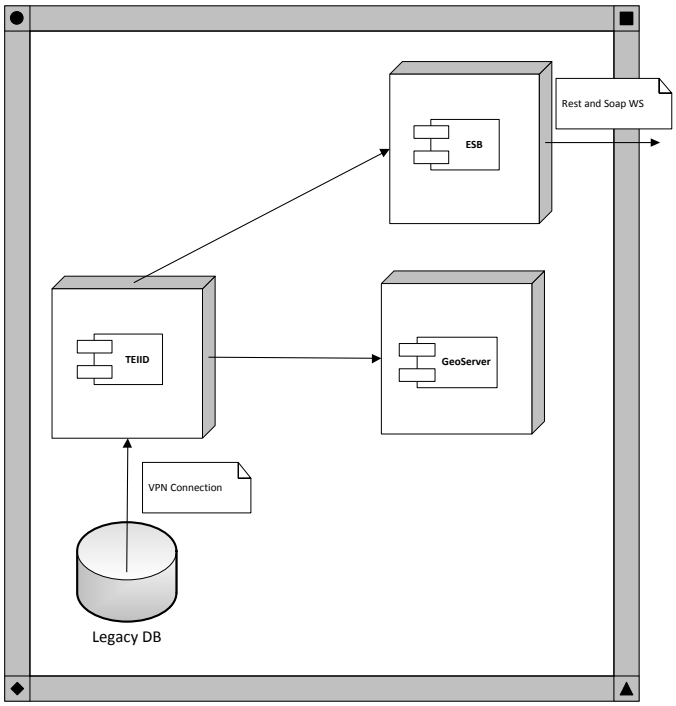


Figure 50: POI Deployment Diagram

2.4.2.2.3 User Manual

POIs will be a mobile application that citizens can use to search for point of interests and also there will be a capability to find closest poi to user.

Use-Case 1: Citizen/Tourist POI Information – This use case identified the actions that a citizen or a tourist can take during POI information retrieval.

Use-Case 2: 3rd Party Development – This use case identifies potential areas of 3rd party development for POI services. This provides actions that 3rd party developers are able to do using the opened data from the platform.

Use-Case 3: System Administration – System is controlled and managed by the system administrators in the PA. This use case provides actions that a system admin can take during the system runtime.

Use-Case 1: Citizen/Tourist POI Information

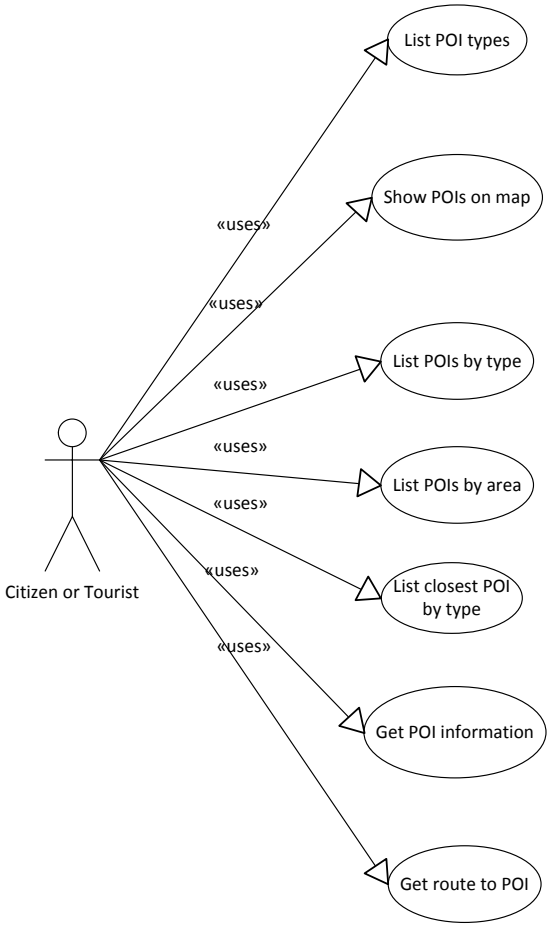


Figure 51: POI Use Case1: Citizen/Tourist POI Information

Actions	**Description**
List POI types	User is able to see to POI types
Show POIs on map	User can see POIs on a map
List POIs by type	User can search for POIs by type
List POIs by area	User can search for POIs by area
List closest POI by type	User can find closest POI for related POI type
Get POI information	User can reach the POI information
Get route to POI	User is able to get route to POI from user location

Use-Case 2: 3rd Party Development

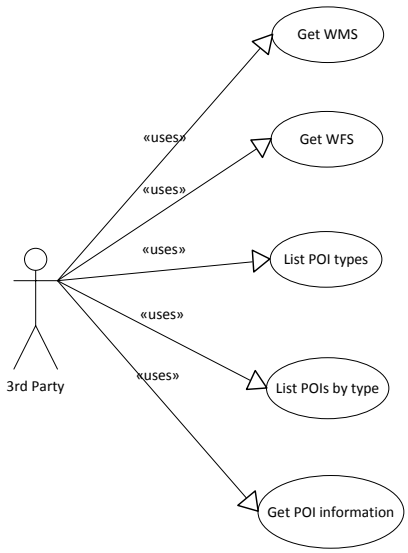


Figure 52: POI Use Case2: 3rd Party Development

Actions	Description
Get WMS	User can reach the maps as images for an area
Get WFS	User can reach the vector spatial data for an area
List POI types	User is able to see to POI types
List POIs by type	User can search for POIs by type
Get POI information	User can reach the POI information

Use-Case 3: System Administration

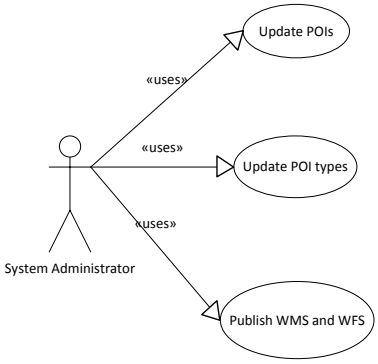


Figure 53: POI Use Case3: System Administration

Actions	Description
Update POIs	Admin is able to create, update or delete a POI
Update POI types	Admin is able to create, update or delete a POI type
Publish WMS and WFS	Admin can publish the WMS and WFS services

2.5 Karlshamn municipality pilot

2.5.1 Reports

Abstract	
The pilot from NetPort Science Park and Karlshamn municipality provides a mobile reporting service that allows citizens and staff to place reports and suggestions on a map. To achieve this we used Open-DAI to virtualize a reports database into a RESTful API, and publish municipality-provided map points as GeoJSON.	
Technologies used on pilot	**Data-sources silos**
• Node.js • MySQL • JSON • Backbone.js • Handlebars.js	• Reports • Points of interest

2.5.1.1 Pilot development

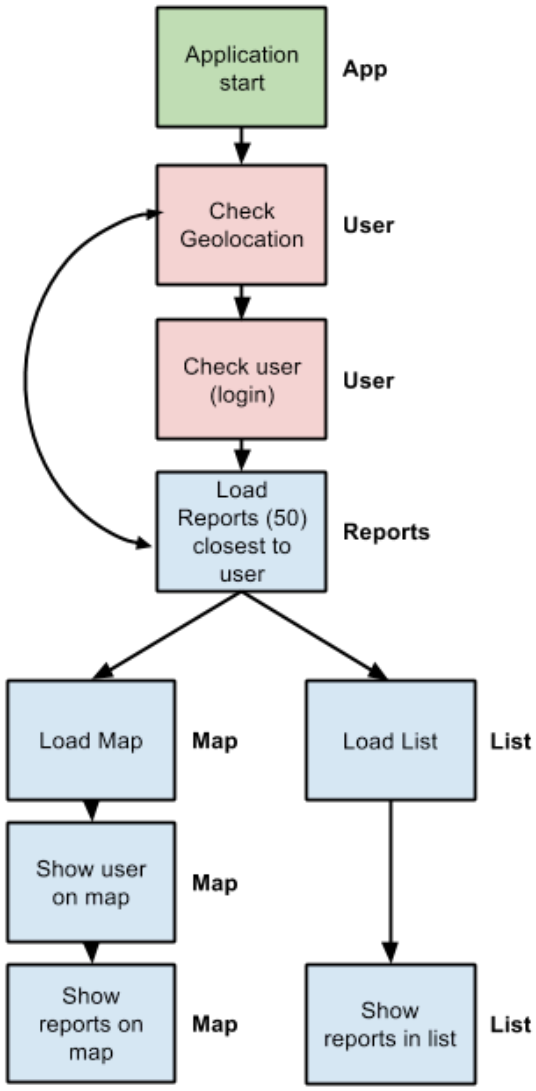


Figure 54: Block Diagram Reports

When the application starts the user is asked to log in using facebook. By using this login method the user doesn't have to register another account just to use our app.

After the login process the application will try to find the user position with the geolocation API. When this is done the map will be loaded and also the closest reports. By loading the application, views and data directly we save the user lots of loading time when navigating within the application.

2.5.1.1.1 Followed development plan

	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18
Database connection										
API development										
Application dev software tests										
Application development										
Application testing										
Application buggfix										
Deploy										

2.5.1.1.2 Development results

The pilot will help the everyday life of citizens I Karlshamn but it is also built to be deployed in any PA which have the same needs of citizens reporting errors on broken equipment. The pilot shows that it is possible for the citizens to be more involved in keeping the city functional and clean. We are trying new technology by building this pilot application and that has resulted in some problems with packages and versions. While the development process was undergoing the software used was updated several times which resulted in bugs and slowed the development process.

The api can be found on the API store:
https://api.opendai.eu/store/apis/info?name=KHReports&version=1.0.0&provider=SwedenAdmin

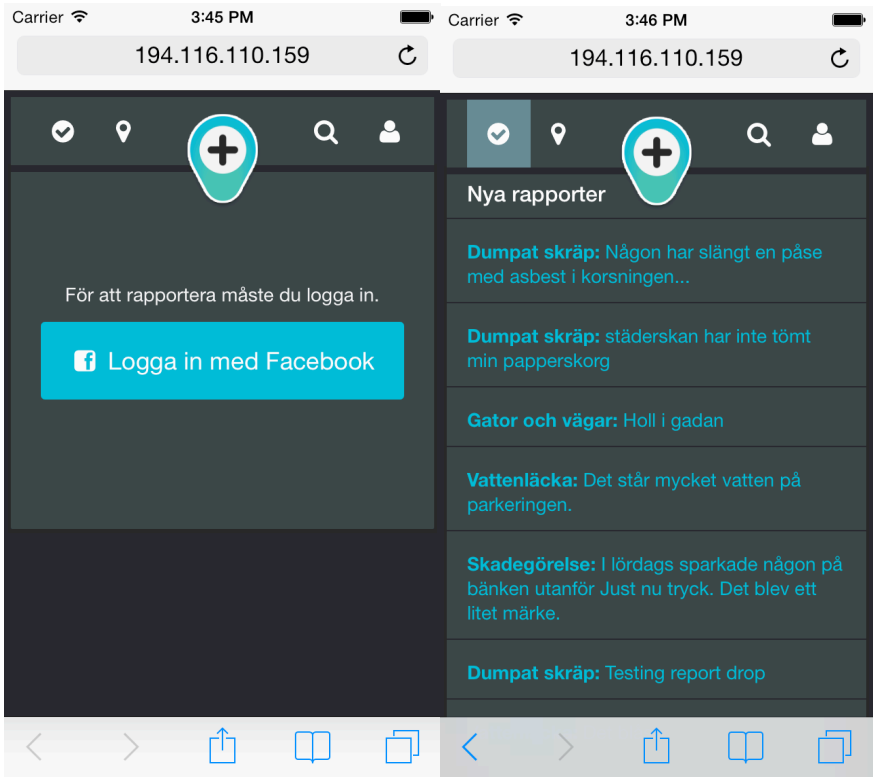

Carrier
3:45 PM
194.116.110.159
För att rapportera måste du logga in.
Logga in med Facebook
Carrier
3:46 PM
194.116.110.159
Nya rapporter
Dumpat skräp: Någon har slängt en påse med asbest i korsningen...
Dumpat skräp: städerskan har inte tömt min papperskorg
Gator och vägar: Holl i gadan
Vattenläcka: Det står mycket vatten på parkeringen.
Skadegörelse: I lördags sparkade någon på bänken utanför Just nu tryck. Det blev ett litet märke.
Dumpat skräp: Testing report drop

Figure 55: Example App (1)

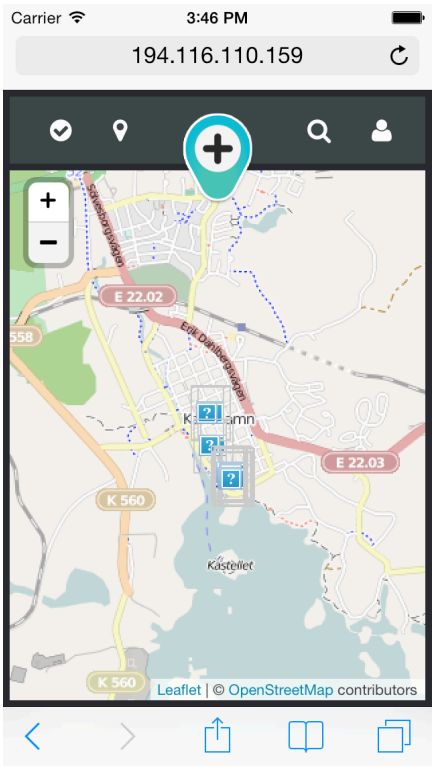


Figure 56: Example App (2)

2.5.1.2 Pilot documentation:

2.5.1.2.1 Pilot code repository

Github repository: https://github.com/open-dai/netport-pilot

In the repository you find the source code for both the server (API) and the application. The server works as an API for the application. It also serves the application as a web server.

2.5.1.2.2 How-to install and deploy

To run the application you have to have node.js and node package manager (npm) installed.

To install first you have to fetch the repository from github. Then you have to run commands to install the dependencies, which are used, by the application and the server. The commands are: npm install for installing the npm dependencies, bower install for installing the bower dependencies. To build and deploy you will have to run the command grunt build. Which will concatenate the files and minimize them to save loading times. You can then run node server.js in the build directory.

The application makes use of Open-DAI TEIID installation for connecting to the MySQL database within Karlshamn municipality. It also makes use of the geoserver for loading static objects like light poles from Karlshamn municipality.

2.5.1.2.3 User Manual

First you have to login with facebook. If the login is successful you will be redirected to the list of reports already created by other users. You will only have a few navigation items in the menu. The one to the left is the list of reports which you are currently watching. Second is all the reports displayed on a map. The third and most important item is the create report which allows you to create a new report. The last ones are settings and user profile.